Isomorphism of intersection and union types

Under consideration for publication in Math. Struct. in Comp. Science

Isomorphism of Intersection and Union Types[†]

Dedicated to Corrado Böhm on the occasion of his 90th Birthday

Mario Coppo, Mariangiola Dezani-Ciancaglini, Ines Margaria, Maddalena Zacchi

Dipartimento di Informatica, Università di Torino

Received 10 June 2015

This paper gives a complete characterisation of type isomorphism definable by terms of a λ-calculus with intersection and union types. Unfortunately, when union is considered the Subject Reduction property does not hold in general. However, it is well known that in the λ-calculus, independently of the considered type system, the isomorphism between two types can be realised only by invertible terms. Notably, all invertible terms are linear terms. In this paper the isomorphism of intersection and union types is investigated using a relevant type system for linear terms enjoying the Subject Reduction property. To characterise type isomorphism, a similarity between types and a type reduction are introduced. Types have a unique normal form with respect to the reduction rules and two types are isomorphic if and only if their normal forms are similar.

1. Introduction

In a calculus with types, two types σ and τ are *isomorphic* if there exist two terms P of type $\sigma \to \tau$ and P' of type $\tau \to \sigma$ such that both their compositions $P \circ P'$ and $P' \circ P$ give the identity (at the proper type). The importance of type isomorphism has been highlighted by Di Cosmo (Di Cosmo, 1995), who noted that the equivalence relation on types induced by the notion of isomorphism allows one to abstract from inessential details in the representation of data in programming languages. Actually types as keys are used in Hoogle (Mitchell, 2008), an Haskell API search engine which allows one to search many standard Haskell libraries by either function name, or by approximate type signature. Neil Mitchell (Mitchell, 2011) remarks that in this application a suitable notion of "closeness" of types is needed, and isomorphism represents one of the possible meanings of type closeness.

The study of type isomorphism started in the 1980s with the aim of finding all the type isomorphisms valid in every model of a given language (Bruce and Longo, 1985). If one looks at this problem choosing as language a λ-calculus with types, one can immediately note the close relation between type isomorphism and λ-term invertibility. Actually, in the untyped λ-calculus a λ-term P is *invertible* if there exists a λ-term P' such that $P \circ P' =_{\beta\eta} P' \circ P =_{\beta\eta} \mathbf{I}$ ($\mathbf{I} = \lambda x.x$). The problem of term invertibility has been extensively studied for the untyped λ-calculus since

[†] This work was partially supported by MIUR Project CINA, ICT COST Action IC1201 BETTY and Ateneo/CSP Project SALT.

1970 and the main result has been the complete characterisation of the invertible λ-terms in λβη-calculus (Dezani-Ciancaglini, 1976): the invertible terms are all and only the *finite hereditary permutators*.

Definition 1.1. [Finite Hereditary Permutator] A *finite hereditary permutator* (FHP for short) is a λ-term of the form (modulo β-conversion)

$$\lambda x y_1 \dots y_n . x(P_1 y_{\pi(1)}) \dots (P_n y_{\pi(n)}) \quad (n \geq 0)$$

where π is a permutation of $1,\dots,n$ (*the permutation of the* FHP), and $P_1,\dots,P_n$ are FHPs.

In the following FHPs are often considered in β-normal form, writing $\lambda x y_1 \dots y_n . x Q_1 \dots Q_n$, where $Q_i \beta\longleftarrow P_i y_{\pi[i]}$ and $y_{\pi[i]}$ is the head variable of Q_i $(1 \leq i \leq n)$.
Note that the identity is trivially an FHP (for $n = 0$). Another example of an FHP is

$$\lambda x y_1 y_2 . x\, y_2\, y_1 =_\beta \lambda x y_1 y_2 . x\,((\lambda z.z)\, y_2)\,((\lambda z.z)\, y_1).$$

It is easy to show that FHPs are closed under composition.

Theorem 1.2. A λ-term is invertible iff it is a finite hereditary permutator.

This result, obtained in the framework of the untyped λ-calculus, has been the basis for studying type isomorphism in different type systems for the λ-calculus. Note that every FHP has, modulo βη-conversion, a unique inverse P^{-1}. It is important to stress that, even if in the type free λ-calculus FHPs are defined modulo βη-conversion (Dezani-Ciancaglini, 1976), in this paper FHPs are considered only modulo β-conversion, because types are invariant neither under η-reduction nor under η-expansion.

Taking into account these properties, the definition of type isomorphism in a λ-calculus with types can be stated as follows:

Definition 1.3 (Type isomorphism). Given a λ-calculus with types, two types σ and τ are isomorphic ($\sigma \approx \tau$) if there exists a pair $< P, P^{-1} >$ of FHPs, inverse of each other, such that $\vdash P : \sigma \to \tau$ and $\vdash P^{-1} : \tau \to \sigma$. The pair $< P, P^{-1} >$ *proves* the isomorphism.

For example the pair $< \lambda x y_1 y_2 y_3 . x y_3 y_1 y_2, \lambda x y_1 y_2 y_3 . x y_2 y_3 y_1 >$ proves the isomorphism

$$\varphi_1 \to \varphi_2 \to \varphi_3 \to \varphi_4 \approx \varphi_3 \to \varphi_1 \to \varphi_2 \to \varphi_4.$$

When $P = P^{-1}$ one can simply write "P proves the isomorphism".

The main approach used to characterise type isomorphism in a given system has been to provide a suitable set of equations and to prove that these equations induce the type isomorphism w.r.t. βη-conversion, i.e. that the types of the FHPs are all and only those induced by the set of equations.

The type isomorphism has been studied first in the simply typed λ-calculus. For this calculus Bruce and Longo proved in (Bruce and Longo, 1985) that only one equation is needed, namely, the **swap** equation:

$$\sigma \to \tau \to \rho \;\approx\; \tau \to \sigma \to \rho$$

Since then, the study has been directed toward richer λ-calculi, obtained from the simply typed λ-calculus in an incremental way, by adding some other type constructors (like product types (Soloviev, 1983; Bruce et al., 1992; Soloviev, 1993)) or by allowing higher-order types (System F (Bruce and Longo, 1985; Di Cosmo, 1995)). In all these type systems the set of equations

grows incrementally in the sense that the set of equations for a typed λ-calculus, obtained by adding a primitive to a given λ-calculus, is an extension of the set of equations of the λ-calculus without that primitive (see (Di Cosmo, 2005)).

In the presence of intersection, this incremental approach does not work, as pointed out in (Dezani-Ciancaglini et al., 2010); in particular with intersection types, the isomorphism is no longer a congruence and type equality in the standard models of intersection types does not entail type isomorphism. These quite unexpected facts required the introduction of a syntactical notion of *type similarity* in order to fully characterise the isomorphic types (Dezani-Ciancaglini et al., 2010).

The study of isomorphism is even harder for type systems with intersection and union types because for these systems, in general, the Subject Reduction property does not hold (Barbanera et al., 1995). Moreover, as in the case of intersection types, the isomorphism of union types is not a congruence and it is not complete for type equality in standard models. For example $\sigma \vee \tau \to \rho$ and $\tau \vee \sigma \to \rho$ are isomorphic, while $(\sigma \vee \tau \to \rho) \vee \varphi$ and $(\tau \vee \sigma \to \rho) \vee \varphi$ are not isomorphic, whenever φ is an atomic type.

The type isomorphism for intersection and union types has been approached in (Coppo et al., 2013), where crucial properties of FHPs have been proved and a set of isomorphism preserving (terminating and confluent) reduction rules for types has been defined. The present paper characterises type isomorphism building on the results given in (Coppo et al., 2013) and using a relation of type similarity which extends the relation defined in (Dezani-Ciancaglini et al., 2010).

Nevertheless similarity is not enough to give a complete characterisation of type isomorphism. Combining the $\to$, $\wedge$ and $\vee$ type operators it is possible to obtain types that, although isomorphic, are not similar. This problem is overcome by exploiting the notion of type normalisation introduced in (Coppo et al., 2013). Two types can then be proved isomorphic if and only if their normal forms are similar. A by-product of this result is the decidability of type isomorphism.

The remainder of this paper is organised as follows: Section 2 introduces a relevant type system for linear terms and shows its interesting properties. Section 3 presents the normalisation rules and recalls important results on type normalisation. Section 4 defines type similarity and gives the main results:

— similar types are isomorphic (*soundness*);
— the normal forms of isomorphic types are similar (*completeness*);
— type isomorphism is decidable.

The proofs of some technical results, here omitted, can be found in (Coppo et al., 2013), where only type isomorphisms that do not require the **swap** equation are studied and hence similarity is not considered.

2. Type assignment system

The abstract syntax of intersection and union types is given by:

$$\sigma \quad ::= \quad \varphi \mid \sigma \to \sigma \mid \sigma \wedge \sigma \mid \sigma \vee \sigma$$

where φ denotes an atomic type and $\sigma, \tau, \rho, \theta, \vartheta, \zeta, \varsigma$ range over arbitrary types. No structural equivalence is assumed between types, for instance $\sigma \vee \tau$ is different from $\tau \vee \sigma$. As usual, paren-

theses are omitted according to the precedence rule "$\vee$ and $\wedge$ over $\rightarrow$", and "$\rightarrow$" associates to the right.

$$(Ax) \qquad x:\sigma \vdash x:\sigma$$

$$(\rightarrow I) \quad \frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x.M:\sigma\rightarrow\tau} \qquad (\rightarrow E) \quad \frac{\Gamma_1 \vdash M:\sigma\rightarrow\tau \quad \Gamma_2 \vdash N:\sigma}{\Gamma_1,\Gamma_2 \vdash MN:\tau}$$

$$(\wedge I) \quad \frac{\Gamma \vdash M:\sigma \;\; \Gamma \vdash M:\tau}{\Gamma \vdash M:\sigma\wedge\tau} \qquad (\wedge E) \quad \frac{\Gamma \vdash M:\sigma\wedge\tau}{\Gamma \vdash M:\sigma} \qquad \frac{\Gamma \vdash M:\sigma\wedge\tau}{\Gamma \vdash M:\tau}$$

$$(\vee I) \quad \frac{\Gamma \vdash M:\sigma}{\Gamma \vdash M:\sigma\vee\tau} \qquad \frac{\Gamma \vdash M:\sigma}{\Gamma \vdash M:\tau\vee\sigma}$$

$$(\vee E) \quad \frac{\Gamma_1, x:\sigma\wedge\theta \vdash M:\rho \quad \Gamma_1, x:\tau\wedge\theta \vdash M:\rho \quad \Gamma_2 \vdash N:(\sigma\vee\tau)\wedge\theta}{\Gamma_1,\Gamma_2 \vdash M[N/x]:\rho}$$

Fig. 1. Typing rules.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[y:\sigma\wedge\rho]}{y:\rho}(\wedge E) \quad \dfrac{[y:\sigma\wedge\rho]}{y:\sigma}(\wedge E)}{y:\rho\wedge\sigma}(\wedge I)}{y:(\rho\wedge\sigma)\vee(\rho\wedge\tau)}(\vee I) \quad \dfrac{\dfrac{\dfrac{[y:\tau\wedge\rho]}{y:\rho}(\wedge E) \quad \dfrac{[y:\tau\wedge\rho]}{y:\tau}(\wedge E)}{y:\rho\wedge\tau}(\wedge I)}{y:(\rho\wedge\sigma)\vee(\rho\wedge\tau)}(\vee I) \quad \dfrac{\dfrac{[x:\rho\wedge(\sigma\vee\tau)]}{x:\sigma\vee\tau}(\wedge E) \quad \dfrac{[x:\rho\wedge(\sigma\vee\tau)]}{x:\rho}(\wedge E)}{x:(\sigma\vee\tau)\wedge\rho}(\wedge I)}{x:(\rho\wedge\sigma)\vee(\rho\wedge\tau)}(\vee E)}{\lambda x.x:\rho\wedge(\sigma\vee\tau)\rightarrow(\rho\wedge\sigma)\vee(\rho\wedge\tau)}(\rightarrow I)$$

Fig. 2. A derivation of $\vdash \lambda x.x:\rho\wedge(\sigma\vee\tau)\rightarrow(\rho\wedge\sigma)\vee(\rho\wedge\tau)$.

Let *the number of top arrows* of a type σ (notation $\uparrow(\sigma)$) be defined as expected:

$$\uparrow(\varphi) = \uparrow(\tau\wedge\rho) = \uparrow(\tau\vee\rho) = 0 \qquad \uparrow(\tau\rightarrow\rho) = \uparrow(\rho)+1$$

The intersection and union type system considered in this paper is a modified version of the basic one introduced in the seminal paper (MacQueen et al., 1986), restricted to linear λ-terms. A λ-term is *linear* if each free or bound variable occurs exactly once in it.

Figure 1 gives the typing rules. As usual *type environments* associate variables with types and contain at most one type for each variable. The type environments are relevant, i.e. they contain only the used premises. When writing Γ_1,Γ_2 one convenes that the sets of variables in Γ_1 and Γ_2 are disjoint.

Some useful admissible rules are:

$$(L) \quad \frac{x:\sigma \vdash x:\tau \quad \Gamma, x:\tau \vdash M:\rho}{\Gamma, x:\sigma \vdash M:\rho} \qquad (C) \quad \frac{\Gamma_1, x:\sigma \vdash M:\tau \quad \Gamma_2 \vdash N:\sigma}{\Gamma_1,\Gamma_2 \vdash M[N/x]:\tau}$$

$$(\vee I') \quad \frac{\Gamma, x:\sigma \vdash M:\rho \quad \Gamma, x:\tau \vdash M:\rho}{\Gamma, x:\sigma\vee\tau \vdash M:\rho} \qquad (\vee E') \quad \frac{\Gamma_1, x:\sigma \vdash M:\rho \quad \Gamma_1, x:\tau \vdash M:\rho \quad \Gamma_2 \vdash N:\sigma\vee\tau}{\Gamma_1,\Gamma_2 \vdash M[N/x]:\rho}$$

The only non-standard rule is $(\vee E)$. This rule takes into account the fact that, in a type system with intersection types, a same variable can be used in a deduction (by applications of the $(\wedge E)$ rule) with different types in different occurrences. It should then be possible, in general, to apply the union elimination only to the type of one of these occurrences. A paradigmatic example is the one in Figure 2, that shows the distributivity of $\wedge$ with respect to $\vee$ (in one direction). In this deduction one occurrence of the variable y is used with type σ in one branch of the $(\vee E)$ rule and with type τ in the other branch. Another occurrence of y is used with type ρ in both branches. Rule $(\vee E)$ is then the right way to formulate union elimination in a type system in which union and intersection interact. It is indeed a generalisation of the $(\vee E')$ rule given in (MacQueen et al., 1986). A last observation is that, being M linear, in an application of the $(\vee E)$ rule, exactly one occurrence of x is replaced in M.

The system of Figure 1 can be extended to non-linear terms simply by erasing, in rules $(\to E)$ and $(\vee E)$, the condition that the type environments need to be disjoint. It is easy to check that this extended system is conservative over the present one. Therefore the types that can be derived for FHPs are the same in the two systems, so the present study of type isomorphism holds for the extended system too.

A fundamental tool to approach the isomorphism problem is the Subject Conversion theorem, proved in (Coppo et al., 2013). The proof of Subject Reduction is based on the classical approach of (Prawitz, 1965) by considering a sequent formulation of the type assignment system and showing cut elimination. This is done in (Barbanera et al., 1995) for a system that differs from the present one for being not relevant, having the universal type and rule $(\vee E')$ instead of $(\vee E)$. The proof of Subject Reduction relies on the fact that, considering only linear terms, cut elimination corresponds to one standard β-reduction, while for arbitrary terms cut elimination corresponds to many standard β-reductions. This fact motivates the failure of Subject Reduction for non-linear terms.

Theorem 2.1 (SC). If $M =_\beta N$ and $\Gamma \vdash M : \sigma$ and N is a linear term, then $\Gamma \vdash N : \sigma$.

In what follows, some key properties (proved in (Coppo et al., 2013)) of the type assignment system of Figure 1 are recalled. Lemma 2.2 concerns types derivable for variables and abstractions. In particular, Point (1) asserts that the inference rules can not change the arrow type that the environment associates with a variable, Point (2) assures that rule $(\to I)$ can be inverted and Point (3) allows to compose by intersections and unions the arrow types derivable for λ-abstractions.

Lemma 2.3 considers the application of a variable to n λ-terms, when the variable in the environment is associated with a type θ such that $\uparrow(\theta) \geq n$. Point (1) relates θ to the types derivable for the application and for the n λ-terms. Point (2) proves the derivability for a variable of the type of the application, assuming for this variable the type obtained from θ by removing the first n top arrows.

Lemma 2.2.

1 If $x : \sigma \to \tau \vdash x : \rho \to \theta$, then $\sigma \to \tau = \rho \to \theta$.
2 If $\Gamma \vdash \lambda x.M : \sigma \to \tau$, then $\Gamma, x : \sigma \vdash M : \tau$.
3 If $\Gamma \vdash \lambda x.M : \sigma \to \rho$ and $\Gamma \vdash \lambda x.M : \tau \to \theta$, then $\Gamma \vdash \lambda x.M : \sigma \wedge \tau \to \rho \wedge \theta$ and $\Gamma \vdash \lambda x.M : \sigma \vee \tau \to \rho \vee \theta$.

In the following, as usual, $FV(M)$ denotes the free variables of M and $\Gamma \restriction FV(M)$ denotes the set of premises in Γ whose subjects are the free variables of M.

Lemma 2.3. Let $\Gamma_x = \Gamma, x:\tau_1 \to \ldots \to \tau_n \to \sigma$ and $\Gamma_x \vdash xM_1 \ldots M_n : \rho$. Then:

1 $\Gamma_x \vdash xM_1 \ldots M_n : \sigma$ and $\Gamma \restriction FV(M_i) \vdash M_i : \tau_i$ for $1 \leq i \leq n$;
2 $y:\sigma \vdash y:\rho$.

In deriving arrow types for FHPs, a key case is when the permutation π of the FHP $P = \lambda x y_1 \ldots y_n.xQ_1 \ldots Q_n$ is such that $\pi(i) \in \{1,\ldots,h\}$ for $1 \leq i \leq h$. In this case P is β-convertible to

$$\lambda x y_1 \ldots y_h.(\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n)(xQ_1 \ldots Q_h),$$

where both $\lambda x y_1 \ldots y_h.xQ_1 \ldots Q_h$ and $\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n$ are FHPs. This is particularly interesting if P "maps" two types with at least h top arrows, let they be $\rho_1 \to \ldots \rho_h \to \sigma$ and $\theta_1 \to \ldots \theta_h \to \tau$. In this case, Theorem 2.5 shows that $\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n$ has type $\sigma \to \tau$ and $\lambda y_{\pi(i)}.Q_i$ has type $\theta_{\pi(i)} \to \rho_i$ for $1 \leq i \leq h$. The proof of this theorem requires an induction on the derivation of

$$x:\rho_1 \to \ldots \rho_h \to \sigma, y_1:\theta_1, \ldots, y_h:\theta_h \vdash (\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n)(xQ_1 \ldots Q_h) : \tau,$$

in which a premise of rule $(\vee E)$ is of the shape

$$x:\rho_1 \to \ldots \rho_h \to \sigma, y_1:\theta_1, \ldots, y_h:\theta_h, t:\vartheta \vdash t(xQ_1 \ldots Q_h) : \tau.$$

For this reason, Lemma 2.4 considers a derivation of

$$\Gamma, x:\rho_1 \to \ldots \rho_h \to \sigma, y_1:\theta_1, \ldots, y_h:\theta_h \vdash M(xQ_1 \ldots Q_h) : \tau,$$

where either M is an FHP and Γ is empty or M is a free variable and it is the only variable occurring in Γ. Appendix A contains the proof of this lemma.

Lemma 2.4. Let $\lambda x y_1 \ldots y_h.xQ_1 \ldots Q_h$ be an FHP with permutation π. If

$$\Gamma, x:\rho_1 \to \ldots \rho_h \to \sigma, y_1:\theta_1, \ldots, y_h:\theta_h \vdash M(xQ_1 \ldots Q_h) : \tau,$$

where M is either an FHP or a free variable, then $\Gamma, z:\sigma \vdash Mz:\tau$ and $y_{\pi(i)} : \theta_{\pi(i)} \vdash Q_i : \rho_i$ for $1 \leq i \leq h$.

Theorem 2.5. Let $\lambda x y_1 \ldots y_n.xQ_1 \ldots Q_n$ be an FHP with permutation π such that $\pi(i) \in \{1,\ldots,h\}$ for $1 \leq i \leq h$. If $x:\rho_1 \to \ldots \to \rho_h \to \sigma, y_1:\theta_1, \ldots, y_h:\theta_h \vdash \lambda y_{h+1} \ldots y_n.xQ_1 \ldots Q_n : \tau$, then

1 $z:\sigma \vdash \lambda y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n : \tau$.
2 $y_{\pi(i)} : \theta_{\pi(i)} \vdash Q_i : \rho_i$ for $1 \leq i \leq h$.

Proof. By Subject Expansion (Theorem 2.1)

$$x:\rho_1 \to \ldots \to \rho_h \to \sigma, y_1:\theta_1, \ldots, y_h:\theta_h \vdash (\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n)(xQ_1 \ldots Q_h) : \tau.$$

This implies $z:\sigma \vdash \lambda y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n : \tau$ and $y_{\pi(i)} : \theta_{\pi(i)} \vdash Q_i : \rho_i$ for $1 \leq i \leq h$, by Lemma 2.4. □

3. Normalisation of types

To investigate type isomorphism, following a common approach (Bruce et al., 1992; Dezani-Ciancaglini et al., 2010), a *normal form* of types is introduced. *Normal type* is short for type in normal form. The notion of normal form is effective, since an algorithm to find the normal form of an arbitrary type is given.

To reduce types, it is useful to consider some basic isomorphisms, which are directly related to standard properties of functional types and to set theoretic properties of union and intersection. It is interesting to remark that all these isomorphisms are provable equalities in the system $\mathbf{B}_+$ of relevant logic (Routley and Meyer, 1972). The following lemma, proved in (Coppo et al., 2013), lists these isomorphisms.

Lemma 3.1. The following isomorphisms hold:

idem.	$\sigma \wedge \sigma \approx \sigma,\ \sigma \vee \sigma \approx \sigma$
comm.	$\sigma \wedge \tau \approx \tau \wedge \sigma,\ \sigma \vee \tau \approx \tau \vee \sigma$
assoc.	$(\sigma \wedge \tau) \wedge \rho \approx \sigma \wedge (\tau \wedge \rho),\ (\sigma \vee \tau) \vee \rho \approx \sigma \vee (\tau \vee \rho)$
dist$\rightarrow \wedge$.	$\sigma \rightarrow \tau \wedge \rho \approx (\sigma \rightarrow \tau) \wedge (\sigma \rightarrow \rho)$
dist$\rightarrow \vee$.	$\sigma \vee \tau \rightarrow \rho \approx (\sigma \rightarrow \rho) \wedge (\tau \rightarrow \rho)$
swap.	$\sigma \rightarrow \tau \rightarrow \rho \approx \tau \rightarrow \sigma \rightarrow \rho$
dist$\wedge\vee$.	$(\sigma \vee \tau) \wedge \rho \approx (\sigma \wedge \rho) \vee (\tau \wedge \rho)$
dist$\vee\wedge$.	$(\sigma \wedge \tau) \vee \rho \approx (\sigma \vee \rho) \wedge (\tau \vee \rho)$

Owing to (**idem**), (**comm**), and (**assoc**) isomorphisms, the types which do occur neither in the left nor in the right- hand-sides of an arrow (*top level* types) can be considered modulo idempotence, commutativity and associativity of $\wedge$ and $\vee$. Then types, at top level, can be written as $\bigwedge_{i\in I}\sigma_i$ and $\bigvee_{i\in I}\sigma_i$ with finite I, where a single atomic or arrow type is seen both as an intersection and as a union (in this case I is a singleton). However, as noted in the introduction, these isomorphisms are not preserved by arbitrary contexts since, for example, $\sigma \vee \tau \rightarrow \rho \approx \tau \vee \sigma \rightarrow \rho$, but $(\sigma \vee \tau \rightarrow \rho) \wedge \varphi$ and $(\tau \vee \sigma \rightarrow \rho) \wedge \varphi$ are not isomorphic.

Normal types are obtained by applying as far as possible a set of normalisation rules. Suitable η-expansions of the identity (dubbed *finite hereditarily identities*, FHIs) show that these rules preserve type isomorphism. The transformations applied to obtain the normal forms are essentially:

— the distribution of intersections over unions or vice versa, in such a way that all types to the right of an arrow are in conjunctive normal form and all types to the left of an arrow are in disjunctive normal form. This is obtained by using the isomorphisms (**dist**$\vee\wedge$) and (**dist**$\wedge\vee$) from left to right (*distribution*);

— the elimination of intersections to the right of arrows and of unions to the left of arrows using the isomorphisms (**dist**$\rightarrow \wedge$) and (**dist**$\rightarrow \vee$) from left to right (*splitting*);

— the elimination of redundant intersections and unions, corresponding to intersections and unions performed on types provably included in one another, as $(\sigma \rightarrow \tau) \wedge (\sigma \vee \rho \rightarrow \tau)$, that can be reduced to $\sigma \vee \rho \rightarrow \tau$; similarly $(\sigma \rightarrow \rho \vee \tau) \wedge (\sigma \rightarrow \tau)$ can be reduced to $\sigma \rightarrow \tau$ (*erasure*);

— the transformation of types at top level in conjunctive normal form using the isomorphism (**dist**$\vee\wedge$) from left to right.

An example of normal type is

$$((\varphi_1 \wedge \varphi_2 \rightarrow \varphi_2 \vee \varphi_3) \vee (\varphi_2 \rightarrow \varphi_5)) \wedge ((\varphi_2 \wedge \varphi_3 \rightarrow \varphi_5) \vee (\varphi_4 \rightarrow \varphi_3 \vee \varphi_5)).$$

Since the normalisation rules have to be applied (whenever possible) also to subtypes, the

(standard) notion of type *context* is needed.

$$\mathcal{C}[\,] ::= [\,] \mid \mathcal{C}[\,] \to \sigma \mid \sigma \to \mathcal{C}[\,] \mid \sigma \wedge \mathcal{C}[\,] \mid \mathcal{C}[\,] \wedge \sigma \mid \sigma \vee \mathcal{C}[\,] \mid \mathcal{C}[\,] \vee \sigma.$$

The possibility of applying transformations to subtypes strongly depends on the context in which they occur. An example of this problem was already given in the Introduction (page 3). Also the types $\sigma \to \tau \to \rho \vee \theta$ and $\sigma \to \tau \to \theta \vee \rho$ are isomorphic in the context $[\,]$, with $\lambda xy_1y_2.xy_1y_2$ proving the isomorphism. The same types are not isomorphic in the context $[\,] \wedge (\varphi \to \varphi)$, because no FHP has type $(\sigma \to \tau \to \rho \vee \theta) \wedge (\varphi \to \varphi) \to (\sigma \to \tau \to \theta \vee \rho) \wedge (\varphi \to \varphi)$.

To realise when a type in a context can be reduced, *paths of type contexts* are useful (Definition 3.3). A path of a context describes which arrows need to be traversed in order to reach the hole, if it is possible, i.e. when there are no atoms on the way. It is needed to assure that no type composed with the type context by means of intersections or unions blocks the transformation. To this aim it is handy to have a notion of *agreement of a type with a path* (Definition 3.2(3)).

An atomic type only agrees with the empty path. An arrow type agrees with a path that goes left (right) if its left(right)-hand-side agrees with the remaining path. An intersection or a union agrees with a path only if all types belonging to the intersection or to the union agree with that path. Notice that if a type agrees with a path, it agrees with all its initial sub-paths.

In paths the symbol $\swarrow$ represents going down to the left of an arrow and the symbol $\searrow$ represents going down to the right of an arrow. As usual ε stands for the empty path.
For distribution rules it is enough to reach the hole, while for splitting rules one more arrow needs to be traversed. So two kinds of paths are useful. They are dubbed d-paths (Definition 3.2(1)) and s-paths (Definition 3.2(2)), being used in distribution and splitting rules, respectively.

Additionally, the agreement of a type with a set of d-paths (Definition 3.2(4)) and the concatenation of d-paths (Definition 3.2(5)) are useful for defining the erasure rules (Definition 3.7).

Definition 3.2.

1 A *d-path* p is a possibly empty string on the alphabet $\{\swarrow, \searrow\}$.
2 An *s-path* p is a d-path followed by $\Box$.
3 The *agreement* of a type σ with a d-path or an s-path p (notation $\sigma \propto \mathsf{p}$) is the smallest relation between types and d-paths (s-paths) such that:

$\sigma \propto \varepsilon$ for all σ;	$\tau \to \rho \propto \Box$ for all τ, ρ;
$\tau \propto \mathsf{p}$ implies $\tau \to \rho \propto_{\swarrow} \mathsf{p}$;	$\rho \propto \mathsf{p}$ implies $\tau \to \rho \propto_{\searrow} \mathsf{p}$;
$\tau \propto \mathsf{p}$ and $\rho \propto \mathsf{p}$ imply $\tau \wedge \rho \propto \mathsf{p}$;	$\tau \propto \mathsf{p}$ and $\rho \propto \mathsf{p}$ imply $\tau \vee \rho \propto \mathsf{p}$.

4 A type σ *agrees* with a set of d-paths $\mathcal{P}$ (notation $\sigma \propto \mathcal{P}$) if it agrees with all the d-paths in $\mathcal{P}$, i.e. $\sigma \propto \mathsf{p}$ for all $\mathsf{p} \in \mathcal{P}$.
5 If p and p' are d-paths, $\mathsf{p} \cdot \mathsf{p}'$ denotes their *concatenation*; if $\mathcal{P}$ is a set of d-paths, $\mathsf{p} \cdot \mathcal{P}$ denotes the set $\{\mathsf{p} \cdot \mathsf{p}' \mid \mathsf{p}' \in \mathcal{P}\} \cup \{\mathsf{p}\}$.

For example the type $\sigma_1 \to (\sigma_2 \to \rho_1 \wedge \rho_2) \wedge (\sigma_3 \vee \sigma_1 \to \tau_1) \to \tau_2$ agrees with the d-path $\searrow\swarrow\swarrow$ and with the s-path $\searrow\swarrow\Box$, while the type $\sigma_1 \to (\sigma_2 \to \rho_1 \wedge \rho_2) \wedge (\sigma_3 \vee \sigma_1 \to \tau_1) \wedge \varphi \to \tau_2$ agrees with the d-path $\searrow\swarrow$ and with the s-path $\searrow\Box$, but it does not agree with the d-path $\searrow\swarrow\swarrow$ and even with the s-path $\searrow\swarrow\Box$, since φ does agree neither with $\swarrow$ nor with $\Box$.

The d-paths and s-paths of contexts can be formalised using the agreement between types and paths.

Definition 3.3. The d-path and the s-path of a type context $\mathcal{C}[\,]$ (notations $d(\mathcal{C}[\,])$ and $s(\mathcal{C}[\,])$, respectively) are defined by induction on $\mathcal{C}[\,]$:

$$d(\mathcal{C}[\,]) = \varepsilon \text{ if } \mathcal{C}[\,] = [\,]; \qquad s(\mathcal{C}[\,]) = \Box \text{ if } \mathcal{C}[\,] = [\,];$$
$$*(\mathcal{C}'[\,]) = \mathsf{p} \text{ implies } *(\mathcal{C}[\,]) = \swarrow \mathsf{p} \text{ if } \mathcal{C}[\,] = \mathcal{C}'[\,] \to \sigma \text{ and } *(\mathcal{C}[\,]) = \searrow \mathsf{p} \text{ if } \mathcal{C}[\,] = \sigma \to \mathcal{C}'[\,];$$
$$\sigma \propto *(\mathcal{C}'[\,]) \text{ implies } *(\mathcal{C}[\,]) = *(\mathcal{C}'[\,]) \text{ if } \mathcal{C}[\,] = \mathcal{C}'[\,] \wedge \sigma \text{ or } \mathcal{C}[\,] = \sigma \wedge \mathcal{C}'[\,] \text{ or } \mathcal{C}[\,] = \mathcal{C}'[\,] \vee \sigma \text{ or } \mathcal{C}[\,] = \sigma \vee \mathcal{C}'[\,],$$

where $*$ holds for both d and s.

For example the d-path and the s-path of the context $\sigma_1 \to [\,] \wedge (\sigma_2 \to \tau_1 \vee \tau_2) \to \tau_2$ are $\searrow\!\!\swarrow$ and $\searrow\!\!\swarrow \Box$, respectively, while the d-path and the s-path of the context $\sigma_1 \to ([\,] \wedge \sigma_2 \to \tau_1) \vee \varphi \to \tau_2$ are undefined, since $\varphi \not\propto \swarrow$ and $\varphi \not\propto \Box$.

In giving the normalisation rules one can consider types in holes modulo idempotence, commutativity and associativity, when the d-paths of contexts are defined. This is assured by the following lemma, that can be easily proved by induction on d-paths.

Lemma 3.4. If $\sigma \approx \tau$ holds by the isomorphisms (**idem**), (**comm**), (**assoc**), and $d(\mathcal{C}[\,])$ is defined, then $\mathcal{C}[\sigma] \approx \mathcal{C}[\tau]$.

Consider, for example, $\tau \vee \rho \approx \rho \vee \tau$. Taking the context $\sigma \to \varphi \vee [\,]$, whose d-path is defined, one has $\sigma \to \varphi \vee \tau \vee \rho \approx \sigma \to \varphi \vee \rho \vee \tau$. On the contrary, taking the context $(\sigma \to \varphi \vee [\,]) \wedge \psi$, whose d-path is not defined, $(\sigma \to \varphi \vee \tau \vee \rho) \wedge \psi$ is not isomorphic to $(\sigma \to \varphi \vee \rho \vee \tau) \wedge \psi$.

Distribution and splitting rules can now be defined.

Definition 3.5 (Distribution and Splitting).

1 The two *distribution rules* are:

$$\mathcal{C}[(\sigma \wedge \tau) \vee \rho] \Longrightarrow \mathcal{C}[(\sigma \vee \rho) \wedge (\tau \vee \rho)] \qquad \text{if } d(\mathcal{C}[\,]) = \varepsilon \text{ or } d(\mathcal{C}[\,]) = \mathsf{p}\cdot \searrow \text{ for some path } \mathsf{p};$$
$$\mathcal{C}[(\sigma \vee \tau) \wedge \rho] \Longrightarrow \mathcal{C}[(\sigma \wedge \rho) \vee (\tau \wedge \rho)] \qquad \text{if } d(\mathcal{C}[\,]) = \mathsf{p}\cdot \swarrow \text{ for some path } \mathsf{p}.$$

2 The two *splitting rules* are:

$$\mathcal{C}[\sigma \to \tau \wedge \rho] \Longrightarrow \mathcal{C}[(\sigma \to \tau) \wedge (\sigma \to \rho)] \text{ if } s(\mathcal{C}[\,]) \text{ is defined;}$$
$$\mathcal{C}[\sigma \vee \tau \to \rho] \Longrightarrow \mathcal{C}[(\sigma \to \rho) \wedge (\tau \to \rho)] \text{ if } s(\mathcal{C}[\,]) \text{ is defined.}$$

For example, by applying first the distribution rules and then the splitting ones, one has:
$(\varphi_1 \vee \varphi_2) \wedge \varphi_3 \to (\varphi_4 \wedge \varphi_5) \vee \varphi_6 \Longrightarrow^+ (\varphi_1 \wedge \varphi_3) \vee (\varphi_2 \wedge \varphi_3) \to (\varphi_4 \vee \varphi_6) \wedge (\varphi_5 \vee \varphi_6) \Longrightarrow^+$
$(\varphi_1 \wedge \varphi_3 \to \varphi_4 \vee \varphi_6) \wedge (\varphi_2 \wedge \varphi_3 \to \varphi_4 \vee \varphi_6) \wedge (\varphi_1 \wedge \varphi_3 \to \varphi_5 \vee \varphi_6) \wedge (\varphi_2 \wedge \varphi_3 \to \varphi_5 \vee \varphi_6)$,
where $\Longrightarrow^+$ is used for the transitive closure of $\Longrightarrow$.

It is useful to distinguish between different kinds of types. So in the following:

— α, β range over atomic and arrow types, formally $\alpha ::= \varphi \mid \sigma \to \sigma$;
— μ, ν, λ range over intersections of atomic and arrow types (*basic intersections*), formally $\mu ::= \alpha \mid \mu \wedge \mu$;
— χ, κ, ι range over unions of atomic and arrow types (*basic unions*), formally $\chi ::= \alpha \mid \chi \vee \chi$.

The rules of Definition 3.5 can be used to transform types into isomorphic types, in particular, by applying the second distribution rule to the left-hand-side and the first distribution rule to the right-hand-side of an arrow type, and then the splitting rules to the obtained type, the achieved arrow type has a basic intersection as left-hand-side and a basic union as right-hand-side.
Ultimately, the types obtained using distribution and splitting rules (dubbed *d-s-normal types*) are such that:

— a d-s-normal arrow type has a d-s-normal basic intersection as left-hand-side and a d-s-normal basic union as right-hand-side;
— a d-s-normal basic intersection (union) "contains" either an atomic type or only d-s-normal arrow types;
— a d-s-normal intersection of basic unions "contains" either an union with an atomic type or only d-s-normal basic unions.

A d-s-normal type, at top level, is a d-s-normal intersection of basic unions.

The erasure rules are defined to eliminate redundant intersections and union types. Since these rules in the normalisation process are applied to d-s-normal types, their definition is based on two preorders, which relate only basic intersections and only basic unions, comparing d-s-normal arrow types.

Definition 3.6 (Preorders on types). The relations $\leq^{\wedge}$ on basic intersections and $\leq^{\vee}$ on basic unions are defined by:

$$\mu \leq^{\wedge} \mu \qquad \chi \leq^{\vee} \chi \qquad \varphi \wedge \mu \leq^{\wedge} \varphi \qquad \varphi \leq^{\vee} \varphi \vee \chi$$
$$\varphi \wedge \mu \wedge \lambda \leq^{\wedge} \varphi \wedge \mu \qquad \varphi \vee \chi \leq^{\vee} \varphi \vee \chi \vee \iota$$
$$\nu_i \leq^{\wedge} \mu_i,\ \chi_i \leq^{\vee} \kappa_i \text{ for all } i \in I \ \Rightarrow \textstyle\bigwedge_{i\in I}(\mu_i \to \chi_i)[\wedge\lambda]^{\dagger} \leq^{\wedge} \bigwedge_{i\in I}(\nu_i \to \kappa_i)$$
$$\nu_i \leq^{\wedge} \mu_i,\ \chi_i \leq^{\vee} \kappa_i \text{ for all } i \in I \ \Rightarrow \textstyle\bigvee_{i\in I}(\mu_i \to \chi_i) \leq^{\vee} \bigvee_{i\in I}(\nu_i \to \kappa_i)[\vee\iota]^{\dagger}$$

The symbol $\leq^{\Diamond}$ stands for either $\leq^{\wedge}$ or $\leq^{\vee}$. It easy to verify that $\alpha \leq^{\wedge} \beta$ if and only if $\alpha \leq^{\vee} \beta$, so comparing two atomic or two arrow types one can write $\alpha \leq^{\Diamond} \beta$. The informal meaning of $\sigma \leq^{\Diamond} \tau$ is that σ is included in τ, in the set-theoretical interpretation of types, where arrow types are controvariant in the domain and covariant in the codomain.

For example the relations $\mu \to \chi \leq^{\Diamond} \mu \wedge \nu \to \chi \vee \kappa$ and $(\mu \wedge \nu \to \chi \vee \kappa) \to \iota \leq^{\Diamond} (\mu \to \chi) \to \iota$ can be derived from $\mu \wedge \nu \leq^{\wedge} \mu$ and $\chi \leq^{\vee} \chi \vee \kappa$. It is easy to show that $\leq^{\wedge}$ and $\leq^{\vee}$ are preorders since transitivity holds. The presence, at top level, of an atomic type on both sides of $\leq^{\Diamond}$ forces atomic and arrow types to be only erased or added. In relating types one can exploit also idempotence. For instance in applying Definition 3.6 to prove $\mu \to \chi \leq^{\wedge} (\mu \wedge \nu \to \chi) \wedge (\mu \to \chi \vee \kappa)$ one must take two copies of $(\mu \to \chi)$ showing $\mu \to \chi \leq^{\Diamond} \mu \wedge \nu \to \chi$ and $\mu \to \chi \leq^{\Diamond} \mu \to \chi \vee \kappa$.

These preorders are crucial for the definition of the erasure rules. In fact some types in an intersection can be erased only if the remaining types are not bigger than the erased ones. Dually some types in a union can be erased only if the remaining types are not smaller than the erased ones. Another necessary condition for erasing types is that the FHIs can "reach" the subtypes in which the types, related by the preorder, differ. In order to realise this fact, one d-path is not

[†] The notation $[\wedge\lambda]$ ($[\vee\iota]$) means that $\wedge\lambda$ ($\vee\iota$) can either occur or not.

$$e(\mu \leq^{\wedge} \mu) = e(\chi \leq^{\vee} \chi) = \{\}$$

$$e(\varphi \wedge \mu \leq^{\wedge} \varphi) = e(\varphi \leq^{\vee} \varphi \vee \chi) = e(\varphi \wedge \mu \wedge \lambda \leq^{\wedge} \varphi \wedge \mu) = e(\varphi \vee \chi \leq^{\vee} \varphi \vee \chi \vee \iota) = \{\varepsilon\}$$

$$\begin{array}{l} e(\bigwedge_{i\in I}(\mu_i \to \chi_i)[\wedge\lambda] \leq^{\wedge} \bigwedge_{i\in I}(\nu_i \to \kappa_i)) \\ e(\bigvee_{i\in I}(\mu_i \to \chi_i) \leq^{\vee} \bigvee_{i\in I}(\nu_i \to \kappa_i)[\vee\iota]) \end{array} = \begin{cases} \{\varepsilon\} & \text{if } \lambda \text{ or } \iota \text{ is present and} \\ & e(\nu_i \leq^{\wedge} \mu_i) = e(\chi_i \leq^{\vee} \kappa_i) = \{\} \text{ for all } i \in I, \\ \bigcup_{i\in I}(\swarrow \cdot e(\nu_i \leq^{\wedge} \mu_i) \cup \searrow \cdot e(\chi_i \leq^{\vee} \kappa_i)) & \text{otherwise} \end{cases}$$

if $\nu_i \leq^{\wedge} \mu_i$, $\chi_i \leq^{\vee} \kappa_i$ for all $i \in I$

Fig. 3. Set of d-paths of a preorder derivation.

enough, since there can be many subtypes in which the types differ, so sets of d-paths are needed. Sets of d-paths are then associated with derivations of preorders between types, so that one can check when a type can be erased in a type context.

The set of d-paths of $\sigma \leq^{\Diamond} \tau$ (notation $e(\sigma \leq^{\Diamond} \tau)$, where e stands for erasure) represents the set of paths that make the points, in which σ and τ differ, accessible. For this reason, $e(\mu \leq^{\wedge} \mu)$ and $e(\chi \leq^{\vee} \chi)$ are defined as the empty set and the sets:

$$e(\varphi \wedge \mu \leq^{\wedge} \varphi),\ e(\varphi \leq^{\vee} \varphi \vee \chi),\ e(\varphi \wedge \mu \wedge \lambda \leq^{\wedge} \varphi \wedge \mu),\ \text{ and } \ e(\varphi \vee \chi \leq^{\vee} \varphi \vee \chi \vee \iota)$$

contain only ε; in the other cases these sets must be built from the sets of the paths associated with the subtypes, using $\swarrow$ and $\searrow$. Figure 3 gives this definition. Notice that $e(\sigma \leq^{\Diamond} \tau) = \{\}$ implies $\sigma = \tau$.

For example $e(\mu \to \chi \leq^{\Diamond} \mu \wedge \nu \to \chi \vee \kappa) = \{\swarrow, \searrow\}$ and

$$e((\mu \wedge \nu \to \chi \vee \kappa) \to \iota \leq^{\Diamond} (\mu \to \chi) \to \iota) = \{\swarrow\swarrow, \swarrow\searrow\}.$$

Finally erasure rules can be defined.

Definition 3.7 (Erasure). The three *erasure rules* are:

$$\begin{array}{ll} \bigwedge_{i\in I} \chi_i \Longrightarrow \bigwedge_{j\in J} \chi_j & \text{if } J \subset I \text{ and } \forall i \in I\ \exists j_i \in J.\ \chi_{j_i} \leq^{\vee} \chi_i \text{ and } \forall j \in J.\ \chi_j \propto \mathcal{P}, \\ & \text{where } \mathcal{P} = \bigcup_{i\in I} e(\chi_{j_i} \leq^{\vee} \chi_i); \\ \mathcal{C}[\bigwedge_{i\in I} \alpha_i] \Longrightarrow \mathcal{C}[\bigwedge_{j\in J} \alpha_j] & \text{if } J \subset I \text{ and } \forall i \in I\ \exists j_i \in J.\ \alpha_{j_i} \leq^{\Diamond} \alpha_i \text{ and } \forall j \in J.\ C[\alpha_j] \propto \mathcal{P}, \\ & \text{where } \mathcal{P} = d(C[\,]) \cdot \bigcup_{i\in I} e(\alpha_{j_i} \leq^{\Diamond} \alpha_i); \\ \mathcal{C}[\bigvee_{i\in I} \alpha_i] \Longrightarrow \mathcal{C}[\bigvee_{j\in J} \alpha_j] & \text{if } J \subset I \text{ and } \forall i \in I\ \exists j_i \in J.\ \alpha_i \leq^{\Diamond} \alpha_{j_i} \text{ and } \forall j \in J.\ C[\alpha_j] \propto \mathcal{P}, \\ & \text{where } \mathcal{P} = d(C[\,]) \cdot \bigcup_{i\in I} e(\alpha_i \leq^{\Diamond} \alpha_{j_i}). \end{array}$$

The first erasure rule is designed to be applied only at top level, i.e. in the empty context.

Examples of type normalisation are:

$((\varphi_1 \wedge \varphi_2 \to \varphi_3 \vee \varphi_2) \vee (\varphi_5 \to \varphi_5)) \wedge ((\varphi_1 \to \varphi_3) \vee (\varphi_5 \to \varphi_5)) \Longrightarrow ((\varphi_1 \to \varphi_3) \vee (\varphi_5 \to \varphi_5))$

by the first erasure rule and

$(\varphi_1 \wedge \varphi_2 \to \varphi_3) \wedge (\varphi_1 \to \varphi_3) \to (\varphi_1 \wedge \varphi_2 \to \varphi_3) \vee (\varphi_1 \to \varphi_3) \Longrightarrow$

$(\varphi_1 \to \varphi_3) \to (\varphi_1 \wedge \varphi_2 \to \varphi_3) \vee (\varphi_1 \to \varphi_3) \Longrightarrow (\varphi_1 \to \varphi_3) \to (\varphi_1 \wedge \varphi_2 \to \varphi_3)$

by the second and third erasure rules.

Reduction rules can create new redexes.

For example the first distribution rule applied to $\sigma \to (\tau \wedge \rho) \vee \theta$ gives $\sigma \to (\tau \vee \theta) \wedge (\rho \vee \theta)$, which can be reduced to $(\sigma \to \tau \vee \theta) \wedge (\sigma \to \rho \vee \theta)$ by the first splitting rule. The second splitting rule applied to $(\sigma \vee \varphi \to \varphi) \wedge (\varphi \wedge \psi \to \varphi)$ gives $(\sigma \to \varphi) \wedge (\varphi \to \varphi) \wedge (\varphi \wedge \psi \to \varphi)$, which can be

reduced to $(\sigma \to \varphi) \wedge (\varphi \to \varphi)$ by the first or second erasure rule. A more interesting example is $(\varphi \wedge (\psi \to \psi) \to \psi) \wedge ((\psi \to \psi) \to \psi) \wedge (((\sigma \vee \tau) \wedge \rho \to \rho) \to \rho)$: this type can only be reduced to $((\psi \to \psi) \to \psi) \wedge (((\sigma \vee \tau) \wedge \rho \to \rho) \to \rho)$ by the first or the second erasure rule and then the second distribution rule becomes applicable.

By applying the erasure rules, it is essential to allow one to remove more than one type in a single step. For example, $(\mu \to \varphi \to \chi) \wedge (\mu \to (\varphi \wedge \nu \to \chi) \vee \psi_1) \wedge (\mu \to (\varphi \wedge \nu \to \chi) \vee \psi_2)$ reduces to $\mu \to \varphi \to \chi$, but it does not reduce to $(\mu \to \varphi \to \chi) \wedge (\mu \to (\varphi \wedge \nu \to \chi) \vee \psi_i)$ for $i = 1$ or $i = 2$. The problem is that $\mu \to (\varphi \wedge \nu \to \chi) \vee \psi_1$ does not agree with $e(\mu \to \varphi \to \chi \leq^{\Diamond} \mu \to (\varphi \wedge \nu \to \chi) \vee \psi_2) = \{\searrow\!\swarrow\}$ and dually exchanging ψ_1 with ψ_2.

The normalisation rules are sound, i.e. $\sigma \Longrightarrow \tau$ implies $\sigma \approx \tau$. These isomorphisms are proved by FHIs, as already said after Lemma 3.1. More precisely for each rule $\sigma \Longrightarrow \tau$ there exist two FHIs $\mathsf{Id}, \mathsf{Id}'$ such that $\vdash \mathsf{Id} : \sigma \to \tau$ and $\vdash \mathsf{Id}' : \tau \to \sigma$. The proof of soundness can be found in (Coppo et al., 2013).

Also existence and unicity of normal forms are shown in (Coppo et al., 2013):

Theorem 3.8 (Normal Forms). The rewriting system of Definitions 3.5 and 3.7 is terminating and confluent.

The normal form of type σ, unique modulo commutativity and associativity, is denoted by $\sigma\downarrow$. The soundness of the normalisation rules implies that each type is isomorphic to its normal form.

Corollary 3.9. $\sigma \approx \sigma\downarrow$.

To sum up, the normal types are d-s-normal types that can not be further reduced by erasure. More precisely they can be characterised as follows:

— a normal arrow type has a normal basic intersection as left-hand-side and a normal basic union as right-hand-side;
— a basic intersection $\bigwedge_{i\in I} \alpha_i$ is normal if it is d-s-normal and for all $J \subset I$ such that $\forall i \in I\ \exists j_i \in J.\ \alpha_{j_i} \leq^{\Diamond} \alpha_i$, there exists $j \in J.\ \alpha_j \not\leq \mathcal{P}$, where $\mathcal{P} = \bigcup_{i\in I} e(\alpha_{j_i} \leq^{\Diamond} \alpha_i)$;
— a basic union $\bigvee_{i\in I} \alpha_i$ is normal if it is d-s-normal and for all $J \subset I$ such that $\forall i \in I\ \exists j_i \in J.\ \alpha_i \leq^{\Diamond} \alpha_{j_i}$, there exists $j \in J.\ \alpha_j \not\leq \mathcal{P}$, where $\mathcal{P} = \bigcup_{i\in I} e(\alpha_i \leq^{\Diamond} \alpha_{j_i})$;
— an intersection of basic unions $\bigwedge_{i\in I} \chi_i$ is normal if it is d-s-normal and for all $J \subset I$ such that $\forall i \in I\ \exists j_i \in J.\ \chi_{j_i} \leq^{\vee} \chi_i$, there exists $j \in J.\ \chi_j \not\leq \mathcal{P}$, where $\mathcal{P} = \bigcup_{i\in I} e(\chi_{j_i} \leq^{\vee} \chi_i)$.

Notice that, since $e(\alpha \leq^{\Diamond} \alpha) = e(\chi \leq^{\vee} \chi) = \{\}$, a normal intersection does not contain identical atomic types or identical arrow types or identical basic unions. For the same reason, a normal union contains neither identical atomic types nor identical arrow types.

The main result proved in (Coppo et al., 2013) is the following theorem, which asserts that two isomorphic normal types have the same top level structure in which atomic and arrow types are pairwise isomorphic.

Theorem 3.10. Let $\bigwedge_{i\in I}(\bigvee_{h\in H_i} \alpha_h^{(i)}) \approx \bigwedge_{j\in J}(\bigvee_{k\in K_j} \beta_k^{(j)})$ and both types be normal. Then $I = J$, $H_i = K_i$ and $\alpha_h^{(i)} \approx \beta_h^{(i)}$ for all $h \in H_i$ and $i \in I$.

Theorem 3.10 does not hold for arbitrary types, since, for example,

$$\varphi_1 \vee \varphi_2 \to \varphi_3 \approx (\varphi_1 \to \varphi_3) \wedge (\varphi_2 \to \varphi_3).$$

4. Soundness and Completeness

This section shows the main result of the paper, i.e. that two types are isomorphic if and only if their normal forms are "similar" (Definition 4.1). The basic aim of the similarity relation is that of formalising isomorphism determined by argument permutations (swap equation). This relation has to take into account the fact that, for two types to be isomorphic, it is not sufficient that they coincide modulo permutations of types in the arrow sequences, as in the case of cartesian products. Indeed the same permutation must be applicable to all types in the corresponding intersections and unions. The key notion of similarity exactly expresses such a condition.
Technically it is handy to introduce the similarity between sequences of types of the same length.

Definition 4.1 (Similarity). The *similarity* relation between two sequences of types $\langle \sigma_1, \ldots, \sigma_m \rangle$, $\langle \tau_1, \ldots, \tau_m \rangle$, written $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_m \rangle$, is the smallest equivalence relation such that:

1 $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \sigma_1, \ldots, \sigma_m \rangle$;

2 if $\langle \sigma_1, \ldots, \sigma_i, \sigma_{i+1}, \ldots . \sigma_{i+n}, \sigma_{i+n+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i, \tau_{i+1}, \ldots . \tau_{i+n}, \tau_{i+n+1}, \ldots, \tau_m \rangle$, then

$$\langle \sigma_1, \ldots, \sigma_i, \textstyle\bigwedge_{j \in \{1,\ldots,n\}} \sigma_{i+j}, \sigma_{i+n+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i, \bigwedge_{j \in \{1,\ldots,n\}} \tau_{i+j}, \tau_{i+n+1}, \ldots, \tau_m \rangle \text{ and}$$
$$\langle \sigma_1, \ldots, \sigma_i, \textstyle\bigvee_{j \in \{1,\ldots,n\}} \sigma_{i+j}, \sigma_{i+n+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i, \bigvee_{j \in \{1,\ldots,n\}} \tau_{i+j}, \tau_{i+n+1}, \ldots, \tau_m \rangle$$

3 if $\langle \sigma_i^{(1)}, \ldots, \sigma_i^{(m)} \rangle \sim \langle \tau_i^{(1)}, \ldots, \tau_i^{(m)} \rangle$ for $1 \leq i \leq n$ and $\langle \rho_1, \ldots, \rho_m \rangle \sim \langle \theta_1, \ldots, \theta_m \rangle$, then

$$\langle \sigma_1^{(1)} \to \ldots \to \sigma_n^{(1)} \to \rho_1, \ldots, \sigma_1^{(m)} \to \ldots \to \sigma_n^{(m)} \to \rho_m \rangle \sim$$
$$\langle \tau_{\pi(1)}^{(1)} \to \ldots \to \tau_{\pi(n)}^{(1)} \to \theta_1, \ldots, \tau_{\pi(1)}^{(m)} \to \ldots \to \tau_{\pi(n)}^{(m)} \to \theta_m \rangle,$$

where π is a permutation of $1, \ldots, n$.

Similarity between types is trivially defined as similarity between unary sequences: $\sigma \sim \tau$ if $\langle \sigma \rangle \sim \langle \tau \rangle$.

In Point(2) of the previous definition, $\bigwedge_{j \in \{1,\ldots,n\}} \sigma_{i+j}$ and $\bigvee_{j \in \{1,\ldots,n\}} \sigma_{i+j}$ are considered modulo commutativity and associativity.

For example, $\sigma \wedge \tau \sim \tau \wedge \sigma$ and $\sigma \to \tau \to \rho \sim \tau \to \sigma \to \rho$ (that represents the basic argument swapping). The isomorphism of the types of the first example is proved by the identity, that one of the types of the second example is proved by the permutator $\lambda x y_1 y_2 . x y_2 y_1$. As a more interesting example, the types:

$\sigma = (\varphi_1 \to \varphi_2 \to \varphi_1 \vee \varphi_2) \wedge (\varphi_3 \to \varphi_4 \to \varphi_3) \to (\varphi_1 \to \varphi_2 \to \varphi_3) \vee (\varphi_3 \to \varphi_4 \to \varphi_5)$ and
$\tau = (\varphi_2 \to \varphi_1 \to \varphi_1 \vee \varphi_2) \wedge (\varphi_4 \to \varphi_3 \to \varphi_3) \to (\varphi_2 \to \varphi_1 \to \varphi_3) \vee (\varphi_4 \to \varphi_3 \to \varphi_5)$

are similar. The FHP proving the isomorphism is $P = \lambda x y_1 y_2 y_3 . x (\lambda z_1 z_2 . y_1 z_2 z_1) y_3 y_2$. Note that both similarity and isomorphism fail if the subtype $(\varphi_2 \to \varphi_1 \to \varphi_3) \vee (\varphi_4 \to \varphi_3 \to \varphi_5)$ of τ is replaced by $(\varphi_1 \to \varphi_2 \to \varphi_3) \vee (\varphi_4 \to \varphi_3 \to \varphi_5)$, since the arguments are permutated in only one of the branches of the $\vee$ operator.

The notion of similarity generalises the corresponding one given in (Dezani-Ciancaglini et al., 2010). A first difference is that of including also union in case 2. More interestingly, when only

intersection is considered, all normal arrow types are of the shape $\mu_1 \to \ldots \to \mu_n \to \varphi$. So in case 3 $\langle \rho_1, \ldots, \rho_m \rangle$ and $\langle \theta_1, \ldots, \theta_m \rangle$ must be the same sequence. When union is added, a normal type is of the shape $\mu_1 \to \ldots \to \mu_n \to \chi$. Since χ is a basic union, also $\langle \rho_1, \ldots, \rho_m \rangle$ and $\langle \theta_1, \ldots, \theta_m \rangle$ must be similar, adding a further case of recursion in the definition of similarity.

The soundness of similarity easily follows from its definition.

Theorem 4.2 (Soundness). If $\sigma \sim \tau$, then $\sigma \approx \tau$.

Proof. By induction on the definition of $\sim$ (Definition 4.1), one shows that $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_m \rangle$ implies that there is a pair $< P, P^{-1} >$ that proves $\sigma_\ell \approx \tau_\ell$, for $1 \leq \ell \leq m$.

(1). $\langle \sigma_1, \ldots, \sigma_m \rangle \sim \langle \sigma_1, \ldots, \sigma_m \rangle$. The identity proves the isomorphism.

(2).$\langle \sigma_1, \ldots, \sigma_i, \bigwedge_{j \in \{1,\ldots,n\}} \sigma_{i+j}, \sigma_{i+n+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i, \bigwedge_{j \in \{1,\ldots,n\}} \tau_{i+j}, \tau_{i+n+1}, \ldots, \tau_m \rangle$ or $\langle \sigma_1, \ldots, \sigma_i, \bigvee_{j \in \{1,\ldots,n\}} \sigma_{i+j}, \sigma_{i+n+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i, \bigvee_{j \in \{1,\ldots,n\}} \tau_{i+j}, \tau_{i+n+1}, \ldots, \tau_m \rangle$ since $\langle \sigma_1, \ldots, \sigma_i, \sigma_{i+1}, \ldots, \sigma_{i+n}, \sigma_{i+n+1}, \ldots, \sigma_m \rangle \sim \langle \tau_1, \ldots, \tau_i, \tau_{i+1}, \ldots, \tau_{i+n}, \tau_{i+n+1}, \ldots, \tau_m \rangle$. By induction there is a pair $< P, P^{-1} >$ that proves $\sigma_\ell \approx \tau_\ell$, for $1 \leq \ell \leq m$. By Lemma 2.2(3), the same pair proves the required isomorphisms.

(3). $$\begin{array}{c} \langle \sigma_1^{(1)} \to \ldots \to \sigma_n^{(1)} \to \rho_1, \ldots, \sigma_1^{(m)} \to \ldots \to \sigma_n^{(m)} \to \rho_m \rangle \sim \\ \langle \tau_{\pi(1)}^{(1)} \to \ldots \to \tau_{\pi(n)}^{(1)} \to \theta_1, \ldots, \tau_{\pi(1)}^{(m)} \to \ldots \to \tau_{\pi(n)}^{(m)} \to \theta_m \rangle \end{array}$$

since $\langle \sigma_i^{(1)}, \ldots, \sigma_i^{(m)} \rangle \sim \langle \tau_i^{(1)}, \ldots, \tau_i^{(m)} \rangle$ for $1 \leq i \leq n$ and $\langle \rho_1, \ldots, \rho_m \rangle \sim \langle \theta_1, \ldots, \theta_m \rangle$. By induction, there are pairs $< P_i, P_i^{-1} >$ such that $\vdash P_i : \sigma_i^{(j)} \to \tau_i^{(j)}$ and $\vdash P_i^{-1} : \tau_i^{(j)} \to \sigma_i^{(j)}$ for $1 \leq j \leq m$, $1 \leq i \leq n$ and a pair $< P_*, P_*^{-1} >$ such that $\vdash P_* : \rho_h \to \theta_h$ and $\vdash P_*^{-1} : \theta_h \to \rho_h$ for $1 \leq h \leq m$. The pair $< P, P^{-1} >$ proving the required isomorphism can then be defined by:

$$\begin{array}{rcl} P & = & \lambda x y_1 \ldots y_n . (P_* (x (P_1^{-1} y_{\pi^{-1}(1)}) \ldots (P_n^{-1} y_{\pi^{-1}(n)}))) \\ P^{-1} & = & \lambda x y_1 \ldots y_n . (P_*^{-1} (x (P_{\pi(1)} y_{\pi(1)}) \ldots (P_{\pi(n)} y_{\pi(n)}))) \end{array}$$

□

Similarity is not complete for arbitrary types, for example

$\varphi_1 \vee \varphi_2 \to \varphi_3 \approx (\varphi_1 \to \varphi_3) \wedge (\varphi_2 \to \varphi_3)$ but $\varphi_1 \vee \varphi_2 \to \varphi_3 \not\approx (\varphi_1 \to \varphi_3) \wedge (\varphi_2 \to \varphi_3)$.
Instead similarity is complete for normal types (Theorem 4.7).

Recall that each FHP P can be written as follows (Definition 1.1):

$$\lambda x y_1 \ldots y_n . x (P_1 y_{\pi(1)}) \ldots (P_n y_{\pi(n)}) \quad (n \geq 0)$$

where π is the permutation of P, and $P_1, \ldots, P_n$ are FHPs. Let n be the *degree* of P and $P_1, \ldots, P_n$ the *components* of P. Notice that if π is the permutation of P, then π^{-1} is the permutation of P^{-1}.

Next lemma shows that two FHPs which are inverses of each other have components pairwise inverses of each other. If two FHPs inverses of each other have different degrees, then the extra components of the FHP with higher degree are FHIs.

An interesting case is when the permutation π of an FHP P is such that $\pi(i) \in \{1, \ldots, h\}$ for $1 \leq i \leq h$, where h is not greater than the degrees of P and P^{-1}. Then by erasing the first h components and the corresponding abstractions from P and P^{-1}, the obtained λ-terms are FHPs inverses of each other. The following lemma formalises this argument.

Lemma 4.3. Let $P = \lambda x y_1 \ldots y_n . x Q_1 \ldots Q_n$ and $P^{-1} = \lambda u t_1 \ldots t_{n'} . u R_1 \ldots R_{n'}$ and let π be the permutation of P. Then:

1 $n \leq n'$ implies that $\lambda y_{\pi(i)}.Q_i$ and $\lambda t_i.R_{\pi(i)}$ are inverses of each other for $1 \leq i \leq n$ and $\lambda t_i.R_{\pi(i)}$ are FHIs for $n+1 \leq i \leq n'$;

2 $h \leq min(n,n')$ and $\pi(i) \in \{1,...,h\}$ for $1 \leq i \leq h$, imply that $\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n$ and $\lambda w t_{h+1} \ldots t_{n'}.wR_{h+1} \ldots R_{n'}$ are inverses of each other.

Proof. (1). Let $P_i = \lambda y_{\pi(i)}.Q_i$ and $P'_{\pi(i)} = \lambda t_i.R_{\pi(i)}$ for $1 \leq i \leq n$. Since

$$P^{-1} \circ P =_\beta \lambda u t_1 \ldots t_{n'}.u(P_1(P'_{\pi(1)}t_1)) \ldots (P_n(P'_{\pi(n)}t_n))R_{n+1} \ldots R_{n'}$$

$\lambda y_{\pi(i)}.Q_i$ and $\lambda t_i.R_{\pi(i)}$ are inverses of each other for $1 \leq i \leq n$ and $\lambda t_{n+1}.R_{n+1}, \ldots, \lambda t_{n'}.R_{n'}$ are FHIs.

(2). If $\pi(i) \in \{1,\ldots,h\}$ for $1 \leq i \leq h$, then $\pi(i) \in \{h+1,\ldots,n\}$ for $h+1 \leq i \leq n$. This implies that $\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n$ and $\lambda w t_{h+1} \ldots t_{n'}.wR_{h+1} \ldots R_{n'}$ are inverses of each other. □

Owing to Theorem 3.10 it is enough to prove completeness only for two isomorphic arrow types. In this case it is crucial to show that the isomorphic types have the same number of top arrows: this is done in Lemma 4.5. The proof of this lemma requires to show that if an FHP of degree n maps a normal type with $h \leq n$ top arrows, then $\pi(i) \in \{1,\ldots,h\}$ for $1 \leq i \leq h$, where π is the permutation of the FHP. This Lemma 4.4, whose proof is the content of Appendix B.

Lemma 4.4. Let P be an FHP with degree n and permutation π. If $\vdash P:\sigma \to \tau$ and σ is a normal type with $\uparrow(\sigma) = h \leq n$, then $\pi(i) \in \{1,\ldots,h\}$ for $1 \leq i \leq h$.

Lemma 4.5. If σ, τ are normal types and $\sigma \approx \tau$, then $\uparrow(\sigma) = \uparrow(\tau)$.

Proof. Let $\mu_1 \to \mu_2 \ldots \to \mu_h \to \chi \approx \nu_1 \to \ldots \to \nu_k \to \kappa$, where $\uparrow(\chi) = \uparrow(\kappa) = 0$. Let $< P, P^{-1} >$ prove this isomorphism, where $P = \lambda x y_1 \ldots y_n.xQ_1 \ldots Q_n$ and $P^{-1} = \lambda u t_1 \ldots t_{n'}.uR_1 \ldots R_{n'}$.

If $n = min(n,n') \leq min(h,k)$, then Lemma 2.2(2) applied to

$$\vdash P:(\mu_1 \to \mu_2 \ldots \to \mu_h \to \chi) \to \nu_1 \to \ldots \to \nu_k \to \kappa$$

gives

$$x:\mu_1 \to \ldots \to \mu_h \to \chi, y_1:\nu_1, \ldots, y_n:\nu_n \vdash xQ_1 \ldots Q_n:\nu_{n+1} \to \ldots \to \nu_k \to \kappa$$

Lemmas 2.3(2) and 2.2(1) require $\mu_{n+1} \to \ldots \to \mu_h \to \chi = \nu_{n+1} \to \ldots \to \nu_k \to \kappa$, which implies $h = k$.

The proof for the case $n' = min(n,n') \leq min(h,k)$ is the same using P^{-1} instead of P.

If $h = min(h,k) \leq min(n,n')$, Lemma 4.4 implies $\pi(i) \in \{1,...,h\}$ for $1 \leq i \leq h$. Theorem 2.5(1) gives

$$z:\chi \vdash \lambda y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n:\nu_{h+1} \to \ldots \to \nu_k \to \kappa \quad \text{and}$$
$$w:\nu_{h+1} \to \ldots \to \nu_k \to \kappa \vdash \lambda t_{h+1} \ldots t_{n'}.wR_{h+1} \ldots R_{n'}:\chi.$$

By Lemma 4.3(2), $\lambda z y_{h+1} \ldots y_n.zQ_{h+1} \ldots Q_n$ and $\lambda w t_{h+1} \ldots t_{n'}.wR_{h+1} \ldots R_{n'}$ are inverses of each other, and therefore $\chi \approx \nu_{h+1} \to \ldots \to \nu_k \to \kappa$, which implies $h = k$, by Theorem 3.10.

The proof for the case $k = min(h,k) \leq min(n,n')$ is the same exchanging P^{-1} with P. □

A last easy lemma follows from the last clause of Definition 4.1 by choosing $m = 1$.

Lemma 4.6. Let π be a permutation of $\{1,\ldots,n\}$. If $\sigma_i \sim \tau_i$ for $1 \leq i \leq n$ and $\rho \sim \theta$, then $\sigma_1 \to \ldots \to \sigma_n \to \rho \sim \tau_{\pi(1)} \to \ldots \to \tau_{\pi(n)} \to \theta$.

Theorem 4.7 (Completeness). If σ, τ are normal types and $\sigma \approx \tau$, then $\sigma \sim \tau$.

Proof. By structural induction on a pair $< P, P^{-1} >$ which proves $\sigma \approx \tau$.
Let $P = \lambda x y_1 \dots y_n . x Q_1 \dots Q_n$ and $P^{-1} = \lambda u t_1 \dots t_{n'} . u R_1 \dots R_{n'}$ with $n \leq n'$ and π be the permutation of P. Lemma 3.10 allows to assume that σ, τ are arrow types and Lemma 4.5 gives $\uparrow(\sigma) = \uparrow(\tau)$. Let $\sigma = \mu_1 \to \mu_2 \dots \to \mu_h \to \chi$ and $\tau = \nu_1 \to \dots \to \nu_h \to \kappa$, where $\uparrow(\chi) = \uparrow(\kappa) = 0$.
If $n' \leq h$ Lemma 2.2(2) gives

$$x : \sigma, y_1 : \nu_1, \dots, y_n : \nu_n \vdash x Q_1 \dots Q_n : \nu_{n+1} \to \dots \to \nu_h \to \kappa \tag{1}$$

$$u : \tau, t_1 : \mu_1, \dots, t_{n'} : \mu_{n'} \vdash u R_1 \dots R_{n'} : \mu_{n'+1} \to \dots \to \mu_h \to \chi \tag{2}$$

The application of Lemmas 2.3(2) and 2.2(1) to (1) implies

$$\mu_{n+1} \dots \to \mu_h \to \chi = \nu_{n+1} \to \dots \to \nu_h \to \kappa.$$

Lemma 2.3(1) and (1) imply $y_{\pi(i)} : \nu_{\pi(i)} \vdash Q_i : \mu_i$ and $(\to I)$ derives $\lambda y_{\pi(i)} . Q_i : \nu_{\pi(i)} \to \mu_i$ for $1 \leq i \leq n$. Lemma 2.3(1) and (2) imply $t_i : \mu_i \vdash R_{\pi(i)} : \nu_{\pi(i)}$ and $(\to I)$ derives $\lambda t_i . R_{\pi(i)} : \mu_i \to \nu_{\pi(i)}$. By Lemma 4.3(1), $\lambda y_{\pi(i)} . Q_i$ and $\lambda t_i . R_{\pi(i)}$ are inverses of each other for $1 \leq i \leq n$. Then by induction, $\mu_i \sim \nu_{\pi(i)}$ for $1 \leq i \leq n$, so Lemma 4.6 gives $\sigma \sim \tau$.

If $n \leq h < n'$, Lemma 2.2(2) gives

$$x : \sigma, y_1 : \nu_1, \dots, y_n : \nu_n \vdash x Q_1 \dots Q_n : \nu_{n+1} \to \dots \to \nu_h \to \kappa \tag{3}$$

$$u : \tau, t_1 : \mu_1, \dots, t_h : \mu_h \vdash \lambda t_{h+1} \dots t_{n'} . u R_1 \dots R_{n'} : \chi \tag{4}$$

The application of Lemmas 2.3(2) and 2.2(1) to (3) implies

$$\mu_{n+1} \dots \to \mu_h \to \chi = \nu_{n+1} \to \dots \to \nu_h \to \kappa.$$

Lemma 2.3(1) and rule $(\to I)$ applied to (3) and to (4) give as in previous case $\lambda y_{\pi(i)} . Q_i : \nu_{\pi(i)} \to \mu_i$ and $\lambda t_i . R_{\pi(i)} : \mu_i \to \nu_{\pi(i)}$ for $1 \leq i \leq n$. By Lemma 4.3(1), $\lambda y_{\pi(i)} . Q_i$ and $\lambda t_i . R_{\pi(i)}$ are inverses of each other for $1 \leq i \leq n$. By induction this implies $\mu_i \sim \nu_{\pi(i)}$ for $1 \leq i \leq n$, so Lemma 4.6 gives $\sigma \sim \tau$.

If $h < n$, Lemma 2.2(2) gives

$$x : \sigma, y_1 : \nu_1, \dots, y_h : \nu_h \vdash \lambda y_{h+1} \dots y_n . x Q_1 \dots Q_n : \kappa \tag{5}$$

$$u : \tau, t_1 : \mu_1, \dots, t_h : \mu_h \vdash \lambda t_{h+1} \dots t_{n'} . u R_1 \dots R_{n'} : \chi \tag{6}$$

By Lemma 4.4, $\pi(i) \in \{1, \dots, h\}$ for $1 \leq i \leq h$. Theorem 2.5(2) and rule $(\to I)$ applied to (5) and to (6) give $\lambda y_{\pi(i)} . Q_i : \nu_{\pi(i)} \to \mu_i$ and $\lambda t_i . R_{\pi(i)} : \mu_i \to \nu_{\pi(i)}$ for $1 \leq i \leq h$. By Lemma 4.3(1), $\lambda y_{\pi(i)} . Q_i$ and $\lambda t_i . R_{\pi(i)}$ are inverses of each other for $1 \leq i \leq h$. By induction this implies $\mu_i \sim \nu_{\pi(i)}$ for $1 \leq i \leq h$.
Theorem 2.5(1) applied to (5) gives $z : \chi \vdash \lambda y_{h+1} \dots y_n . z Q_{h+1} \dots Q_n : \kappa$ and $(\to I)$ derives

$$\vdash \lambda z y_{h+1} \dots y_n . z Q_{h+1} \dots Q_n : \chi \to \kappa.$$

Theorem 2.5(1) applied to (6) gives $w : \kappa \vdash \lambda t_{h+1} \dots t_{n'} . w R_{h+1} \dots R_{n'} : \chi$ and $(\to I)$ derives

$$\vdash \lambda w t_{h+1} \dots t_{n'} . w R_{h+1} \dots R_{n'} : \kappa \to \chi.$$

By Lemma 4.3(2), $\lambda z y_{h+1} \dots y_n . z Q_{h+1} \dots Q_n$ and $\lambda w t_{h+1} \dots t_{n'} . w R_{h+1} \dots R_{n'}$ are inverses of each other. By induction this implies $\chi \sim \kappa$, so $\sigma \sim \tau$, by Lemma 4.6. □

The result of the present paper is summarised in the following theorem.

Theorem 4.8 (Main). Two types are isomorphic if and only if their normal forms are similar.

Proof. Recall that a type is isomorphic to its normal form (Corollary 3.9).

1 ($\Rightarrow$) If $\sigma \approx \tau$, then $\sigma{\downarrow} \approx \sigma \approx \tau \approx \tau{\downarrow}$, from which, by the Completeness Theorem (Theorem 4.7), $\sigma{\downarrow} \sim \tau{\downarrow}$;

2 ($\Leftarrow$) If $\sigma{\downarrow} \sim \tau{\downarrow}$, then, by the Soundness Theorem (Theorem 4.2), $\sigma{\downarrow} \approx \tau{\downarrow}$, hence: $\sigma \approx \sigma{\downarrow} \approx \tau{\downarrow} \approx \tau$, i.e., $\sigma \approx \tau$.

□

A direct consequence of the Main Theorem is the decidability of type isomorphism.

Theorem 4.9. Type isomorphism in the system with intersection and union types is decidable.

Proof. By Theorem 4.8, for deciding if two types are isomorphic it is sufficient to check if their normal forms are similar. These normal forms can be computed owing to the fact that the normalisation rules are terminating and confluent. By Definition 4.1, two types are similar when the unary sequences built by these types are similar, then it enough to show that similarity of type sequences is decidable. This is done by induction on the total number of symbols in the types which occur in the two sequences. Let the sequences be $\langle \sigma_1, \ldots, \sigma_m \rangle$ and $\langle \tau_1, \ldots, \tau_m \rangle$. There are the following cases (leaving out the symmetric ones):

1 If one of the σ_i is an atomic type, then one must have $\sigma_i = \tau_i$ for $1 \leq i \leq m$ (base step).

2 If one of the σ_i is an intersection $\bigwedge_{j\in\{1,\ldots,n\}} \chi_j$ (case 2 of Definition 4.1), then the corresponding τ_i must be of the form $\bigwedge_{j\in\{1,\ldots,n\}} \kappa_j$ and there must exist a permutation π of $\{1,\ldots,n\}$ such that the two sequences
$$\langle \sigma_1, \ldots, \sigma_{i-1}, \chi_1, \ldots, \chi_n, \sigma_{i+1}, \ldots, \sigma_m \rangle \text{ and } \langle \tau_1, \ldots, \tau_{i-1}, \kappa_{\pi(1)}, \ldots, \kappa_{\pi(n)}, \tau_{i+1}, \ldots, \tau_m \rangle$$
are similar. Note that the number of permutations is finite and all sequences to be checked have types with lower numbers of symbols.

3 If one of the σ_i is a union $\bigvee_{j\in\{1,\ldots,n\}} \alpha_j$ (case 2 of Definition 4.1), the proof is as in case 2.

4 If all types in the sequences are arrow types, let they be
$$\langle \sigma_1^{(1)} \to \ldots \to \sigma_n^{(1)} \to \rho_1, \ldots, \sigma_1^{(m)} \to \ldots \to \sigma_n^{(m)} \to \rho_m \rangle \text{ and}$$
$$\langle \tau_1^{(1)} \to \ldots \to \tau_n^{(1)} \to \theta_1, \ldots, \tau_1^{(m)} \to \ldots \to \tau_n^{(m)} \to \theta_m \rangle,$$
for some $n > 0$, where $\uparrow(\rho_i) = 0$ for some i $(1 \leq i \leq m)$ (case 3 of Definition 4.1). Then there must exist a permutation π of $\{1,\ldots,n\}$ such that the following similarities hold: $\langle \rho_1, \ldots, \rho_m \rangle \sim \langle \theta_1, \ldots, \theta_m \rangle$ and $\langle \sigma_i^{(1)}, \ldots, \sigma_i^{(m)} \rangle \sim \langle \tau_{\pi(i)}^{(1)}, \ldots, \tau_{\pi(i)}^{(m)} \rangle$ for $1 \leq i \leq n$.

□

Note that in the system of (Dezani-Ciancaglini et al., 2010), in which only intersection types are considered, decidability is a rather immediate consequence of the decidability of type assignment for λ-terms in normal form, proved in (Ronchi Della Rocca, 1988). This result does not seem immediately extensible to intersection and union types owing to the presence of rule $(\vee E)$.

5. Conclusion

In (Dezani-Ciancaglini et al., 2010) is observed that isomorphism of intersection types is not a congruence and it is not entailed by type equality in standard models. Not surprisingly, also isomorphism of intersection and union types is not a congruence and does not respect semantic equality. For example, the types $\sigma \to \tau \vee \rho$ and $(\sigma \to \tau \vee \rho) \vee (\sigma \to \tau)$ are isomorphic and have the same meaning. On the contrary, types obtained by adding a seemly innocent intersection (or union) with an atomic type, for example

$$(\sigma \to \tau \vee \rho) \wedge \varphi \text{ and } (\sigma \to \tau \vee \rho) \vee (\sigma \to \tau) \wedge \varphi,$$

are semantically equivalent but not isomorphic. It remains an open question to find an universal model for this isomorphism.

In (Coppo et al., 2014b; Coppo et al., 2014a) each atomic type is defined equivalent to an arrow type ("functional" type). The type isomorphism in the resulting type system properly includes the present one, since for example $(\sigma \vee \tau \to \rho) \wedge \varphi$ and $(\tau \vee \sigma \to \rho) \wedge \varphi$ become isomorphic. A characterisation of isomorphism for "functional" intersection types is the result of (Coppo et al., 2014b), while only a condition assuring type isomorphism for "functional" intersection and union types is given in (Coppo et al., 2014a). The authors conjecture that this condition is also necessary; the proof of completeness is left as future work.

The importance of type isomorphism has recently highlighted by Díaz-Caro and Dowek. They pointed out in (Díaz-Caro and Dowek, 2015) that in typed lambda-calculus, in programming languages, and in proof theory, isomorphic types are often identified. For example, the definitionally equivalent types are identified in Martin-Löf's type theory and in the Calculus of Constructions. To distinguish isomorphic types can entail useless drawbacks; for instance, if a library contains a function of type $\sigma \wedge \tau \to \rho$, a request on a function of type $\tau \wedge \sigma \to \rho$ will not have success. For this reason (Díaz-Caro and Dowek, 2015) proposes a type system in which λ-terms getting a type have also all types isomorphic to it.

Another interesting direction of investigation is suggested by the a method for elaborating programs with intersection and union types proposed by Joshua Dunfield in (Dunfield, 2014). In this paper intersections are elaborated into products, and unions into sums. The resulting programs have no intersections and no unions, and can be compiled using conventional means - any SML compiler will do. Remarkably, the isomorphism of product and sum types is a congruence and it is not finitely axiomatisable (Fiore et al., 2006), while the isomorphism of intersection and union types is not a congruence and it is characterised by the present notion of similarity. As future work we plan to investigate how this elaboration relates to isomorphism.

Acknowledgments. We would like to thank the anonymous referees for their detailed remarks and helpful comments.

References

Barbanera, F., Dezani-Ciancaglini, M., and de'Liguoro, U. (1995). Intersection and union types: Syntax and semantics. *Information and Computation*, 119:202–230.

Bruce, K., Di Cosmo, R., and Longo, G. (1992). Provable isomorphisms of types. *Mathematical Structures in Computer Science*, 2(2):231–247.

Bruce, K. and Longo, G. (1985). Provable isomorphisms and domain equations in models of typed languages. In Sedgewick, R., editor, *STOC'85*, pages 263 – 272. ACM Press.

Coppo, M., Dezani-Ciancaglini, M., Margaria, I., and Zacchi, M. (2013). Towards isomorphism of intersection and union types. In Graham-Lengrand, S. and Paolini, L., editors, *ITRS'12*, volume 121 of *EPTCS*, pages 58 – 80.

Coppo, M., Dezani-Ciancaglini, M., Margaria, I., and Zacchi, M. (2014a). Isomorphism of "functional" intersection and union types. In Reholf, J., editor, *ITRS'14*, EPTCS. To appear.

Coppo, M., Dezani-Ciancaglini, M., Margaria, I., and Zacchi, M. (2014b). Isomorphism of "functional" intersection types. In Matthes, R. and Schubert, A., editors, *Types'13*, volume 26, pages 129–149. LIPIcs.

Dezani-Ciancaglini, M. (1976). Characterization of normal forms possessing an inverse in the $\lambda\beta\eta$-calculus. *Theoretical Computer Science*, 2(3):323–337.

Dezani-Ciancaglini, M., Cosmo, R. D., Giovannetti, E., and Tatsuta, M. (2010). On isomorphisms of intersection types. *ACM Transactions on Computational Logic*, 11(4):1–22.

Di Cosmo, R. (1995). Second order isomorphic types. A proof theoretic study on second order λ-calculus with surjective pairing and terminal object. *Information and Computation*, 119(2):176–201.

Di Cosmo, R. (2005). A short survey of isomorphisms of types. *Mathematical Structures in Computer Science*, 15:825–838.

Díaz-Caro, A. and Dowek, G. (2015). Simply typed lambda-calculus modulo type isomorphisms. *Theoretical Computer Science*. To appear.

Dunfield, J. (2014). Elaborating intersection and union types. *Journal of Functional Programmming*, 24(2-3):133–165.

Fiore, M., Di Cosmo, R., and Balat, V. (2006). Remarks on isomorphisms in typed lambda calculi with empty and sum types. *Annals of Pure and Applied Logic*, 141(1–2):35–50.

MacQueen, D., Plotkin, G., and Sethi, R. (1986). An ideal model for recursive polymorphic types. *Information and Control*, 71(1-2):95–130.

Mitchell, N. (2008). Hoogle overview. *The Monad.Reader*, 12:27–35.

Mitchell, N. (2011). Hoogle: Finding functions from types. Invited Presentation from TFP 2011.

Prawitz, D. (1965). *Natural Deduction*. Almqvist & Wiksell.

Ronchi Della Rocca, S. (1988). Principal type scheme and unification for intersection type discipline. *Theoretical Computer Science*, 59(1-2):1–29.

Routley, R. and Meyer, R. K. (1972). The semantics of entailment III. *Journal of Philosophical Logic*, 1:192–208.

Soloviev, S. (1983). The category of finite sets and cartesian closed categories. *Journal of Soviet Mathematics*, 22(3):1387–1400. English translation of the original paper in russian published in Zapiski Nauchyn Seminarov LOMI, v.105, 1981.

Soloviev, S. (1993). A complete axiom system for isomorphism of types in closed categories. In Voronkov, A., editor, *LPAR'93*, volume 698 of *LNCS*, pages 360–371. Springer-Verlag.

Appendix A.

A preliminary lemma is useful for both Appendices.

Lemma A.1.

1 If $x:\sigma\to\tau\vdash x:(\rho\vee\theta)\wedge\vartheta$, then either $x:\sigma\to\tau\vdash x:\rho\wedge\vartheta$ or $x:\sigma\to\tau\vdash x:\theta\wedge\vartheta$.

2 Let χ be a union of atomic and arrow types pairwise different. Then $x:\chi\vdash x:\kappa$ implies either $\kappa=\chi$ or $\kappa=\chi\vee\iota$ for some type ι.

Note that Point (2) of previous lemma holds only under the given condition on type χ, since for example $x:(\varphi\to\varphi)\vee(\varphi\to\varphi)\vdash x:\varphi\to\varphi$.

Proof of Lemma 2.4

A stronger statement is proved:

Let $\lambda x y_1 \dots y_h.xQ_1 \dots Q_h$ *be an* FHP *with permutation* π. *If* $x:\rho_1 \to \dots \rho_h \to \sigma \vdash x:\vartheta$ *and* $\Gamma, x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash M(xQ_1 \dots Q_h):\tau$, *where* M *is either an* FHP *or a free variable, then* $\Gamma, z:\sigma \vdash Mz:\tau$ *and* $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ *for* $1 \leq i \leq h$.

The proof is by induction on the derivation of $\Gamma, x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash M(xQ_1 \dots Q_h):\tau$. By construction Γ is either empty or it contains the unique variable M.

If the last applied rule is $(\to E)$:

$$\frac{\Gamma \vdash M:\zeta \to \tau \qquad x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash xQ_1 \dots Q_h:\zeta}{\Gamma, x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash M(xQ_1 \dots Q_h):\tau}$$

rule (L) applied to $x:\rho_1 \to \dots \rho_h \to \sigma \vdash x:\vartheta$ and to the second premise derives

$$x:\rho_1 \to \dots \rho_h \to \sigma, y_1:\theta_1, \dots, y_h:\theta_h \vdash xQ_1 \dots Q_h:\zeta \tag{7}$$

Lemma 2.3(1) applied to (7) gives $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ for $1 \leq i \leq h$. Lemma 2.3(2) applied to (7) gives $z:\sigma \vdash z:\zeta$, so the first premise implies $\Gamma, z:\sigma \vdash Mz:\tau$, using rule $(\to E)$.

If the last applied rule is $(\wedge I)$, $(\wedge E)$ or $(\vee I)$ the proof by induction is easy.

If the last applied rule is $(\vee E)$ there are six possible cases according to the shape of the subjects of the three premises.

In the first case:

$$\frac{t:\zeta_1 \wedge \varsigma \vdash t:\tau \qquad t:\zeta_2 \wedge \varsigma \vdash t:\tau \qquad \Gamma, x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash M(xQ_1 \dots Q_h):(\zeta_1 \vee \zeta_2) \wedge \varsigma}{\Gamma, x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash M(xQ_1 \dots Q_h):\tau}$$

Induction on the third premise gives $\Gamma, z:\rho \vdash Mz:(\zeta_1 \vee \zeta_2) \wedge \varsigma$ and $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ for $1 \leq i \leq h$. The application of rule $(\vee E)$ to the first two premises and to $\Gamma, z:\rho \vdash Mz:(\zeta_1 \vee \zeta_2) \wedge \varsigma$ derives $\Gamma, z:\sigma \vdash Mz:\tau$.

In the second case:

$$\frac{\Gamma', t:\zeta_1 \wedge \varsigma \vdash t(xQ_1 \dots Q_h):\tau \qquad \Gamma', t:\zeta_2 \wedge \varsigma \vdash t(xQ_1 \dots Q_h):\tau \qquad \Gamma \vdash M:(\zeta_1 \vee \zeta_2) \wedge \varsigma}{\Gamma, x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash M(xQ_1 \dots Q_h):\tau}$$

where $\Gamma' = x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h$. Induction on the first two premises gives $t:\zeta_1 \wedge \varsigma, z:\sigma \vdash tz:\tau$ and $t:\zeta_2 \wedge \varsigma, z:\sigma \vdash tz:\tau$ and $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ for $1 \leq i \leq h$. The application of rule $(\vee E)$ to $t:\zeta_1 \wedge \varsigma, z:\sigma \vdash tz:\tau$ and $t:\zeta_2 \wedge \varsigma, z:\sigma \vdash tz:\tau$ and to the third premise derives $\Gamma, z:\sigma \vdash Mz:\tau$.

In the third case:

$$\frac{\Gamma, t:\zeta_1 \wedge \varsigma \vdash Mt:\tau \qquad \Gamma, t:\zeta_2 \wedge \varsigma \vdash Mt:\tau \qquad x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash xQ_1 \dots Q_h:(\zeta_1 \vee \zeta_2) \wedge \varsigma}{\Gamma, x:\vartheta, y_1:\theta_1, \dots, y_h:\theta_h \vdash M(xQ_1 \dots Q_h):\tau}$$

Rule (L) applied to $x:\rho_1 \to \dots \rho_h \to \sigma \vdash x:\vartheta$ and to the third premise derives

$$x:\rho_1 \to \dots \rho_h \to \sigma, y_1:\theta_1, \dots, y_h:\theta_h \vdash xQ_1 \dots Q_h:(\zeta_1 \vee \zeta_2) \wedge \varsigma \tag{8}$$

Lemma 2.3(1) applied to (8) gives $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ for $1 \leq i \leq h$. Lemma 2.3(2) applied to (8) gives $z:\sigma \vdash z:(\zeta_1 \vee \zeta_2) \wedge \varsigma$, which together with the first two premises implies $\Gamma, z:\sigma \vdash Mz:\tau$ by using rule $(\vee E)$.

In the fourth case:

$$\frac{\Gamma', t:\zeta_1 \wedge \varsigma \vdash M(tQ_{k+1}\dots Q_h):\tau \qquad \Gamma', t:\zeta_2 \wedge \varsigma \vdash M(tQ_{k+1}\dots Q_h):\tau \quad x:\vartheta, y_1:\theta_1,\dots,y_k:\theta_k \vdash xQ_1\dots Q_k:(\zeta_1\vee\zeta_2)\wedge\varsigma}{\Gamma, x:\vartheta, y_1:\theta_1,\dots,y_h:\theta_h \vdash M(xQ_1\dots Q_h):\tau}$$

where $1 \le k < h$ and $\Gamma' = \Gamma, y_{k+1}:\theta_1,\dots,y_h:\theta_h$. Rule (L) applied to $x:\rho_1\to\dots\rho_h\to\sigma \vdash x:\vartheta$ and to the third premise derives

$$x:\rho_1\to\dots\rho_h\to\sigma, y_1:\theta_1,\dots,y_k:\theta_k \vdash xQ_1\dots Q_k:(\zeta_1\vee\zeta_2)\wedge\varsigma. \tag{9}$$

Lemma 2.3(1) applied to (9) gives $x:\rho_1\to\dots\rho_h\to\sigma \vdash xQ_1\dots Q_k:\rho_{k+1}\to\dots\rho_h\to\sigma$ and $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ for $1\le i\le k$.
Lemma 2.3(2) applied to (9) gives $z:\rho_{k+1}\to\dots\rho_h\to\sigma \vdash z:(\zeta_1\vee\zeta_2)\wedge\varsigma$. By Lemma A.1(1) one has either $z:\rho_{k+1}\to\dots\rho_h\to\sigma \vdash z:\zeta_1\wedge\varsigma$ or $z:\rho_{k+1}\to\dots\rho_h\to\sigma \vdash z:\zeta_2\wedge\varsigma$. Induction on the first or on the second premise implies $\Gamma, z:\sigma \vdash Mz:\tau$ and $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ for $k+1\le i\le h$.

In the fifth case:

$$\frac{\Gamma', t:\zeta_1 \wedge \varsigma \vdash M(xQ_1\dots Q_{k-1}tQ_{k+1}\dots Q_h):\tau \quad \Gamma', t:\zeta_2 \wedge \varsigma \vdash M(xQ_1\dots Q_{k-1}tQ_{k+1}\dots Q_h):\tau \quad y_l:\theta_l \vdash Q_k:(\zeta_1\vee\zeta_2)\wedge\varsigma}{\Gamma, x:\vartheta, y_1:\theta_1,\dots,y_h:\theta_h \vdash M(xQ_1\dots Q_h):\tau}$$

where $l=\pi(k)$ and $\Gamma' = \Gamma, x:\vartheta, y_1:\theta_1,\dots,y_{l-1}:\theta_{l-1}, y_{l+1}:\theta_{l+1},\dots,y_h:\theta_h$. Notice that

$$\lambda x y_1\dots y_{l-1} t y_{l+1}\dots y_h.xQ_1\dots Q_{k-1}tQ_{k+1}\dots Q_h$$

is an FHP, so by induction on the first two premises $\Gamma, z:\sigma \vdash Mz:\tau$ and $y_{\pi(i)}:\theta_{\pi(i)} \vdash Q_i:\rho_i$ for $1\le i\le h, i\ne k$ and $t:\zeta_1\wedge\varsigma\vdash t:\rho_k$ and $t:\zeta_2\wedge\varsigma\vdash t:\rho_k$. The application of $(\vee E)$ to $t:\zeta_1\wedge\varsigma\vdash t:\rho_k$ and $t:\zeta_2\wedge\varsigma\vdash t:\rho_k$ and to the third premise gives $y_l:\theta_l\vdash Q_k:\rho_k$.

In the sixth case:

$$\frac{\Gamma', t:\zeta_1 \wedge \varsigma \vdash M(xQ_1\dots Q_{k-1}Q'_kQ_{k+1}\dots Q_h):\tau \quad \Gamma', t:\zeta_2 \wedge \varsigma \vdash M(xQ_1\dots Q_{k-1}Q'_kQ_{k+1}\dots Q_h):\tau \quad y_l:\theta_l \vdash y_l:(\zeta_1\vee\zeta_2)\wedge\varsigma}{\Gamma, x:\vartheta, y_1:\theta_1,\dots,y_h:\theta_h \vdash M(xQ_1\dots Q_h):\tau}$$

where $l=\pi(k)$ and $Q'_k = Q_k[t/y_l]$ and $\Gamma' = \Gamma, x:\vartheta, y_1:\theta_1,\dots,y_{l-1}:\theta_{l-1}, y_{l+1}:\theta_{l+1},\dots,y_h:\theta_h$. Notice that $\lambda x y_1\dots y_{l-1} t y_{l+1}\dots y_h.xQ_1\dots Q_{k-1}Q'_kQ_{k+1}\dots Q_h$ is an FHP, so by induction on the first two premises $\Gamma, z:\sigma\vdash Mz:\tau$ and $y_{\pi(i)}:\theta_{\pi(i)}\vdash Q_i:\rho_i$ for $1\le i\le h, i\ne k$ and $t:\zeta_1\wedge\varsigma\vdash Q'_k:\rho_k$ and $t:\zeta_2\wedge\varsigma\vdash Q'_k:\rho_k$. The application of $(\vee E)$ to $t:\zeta_1\wedge\varsigma\vdash Q'_k:\rho_k$ and $t:\zeta_2\wedge\varsigma\vdash Q'_k:\rho_k$ and to the third premise gives $y_l:\theta_l\vdash Q_k:\rho_k$.

Appendix B.

Proof of Lemma 4.4

A stronger statement is proved:

Let $\lambda x y_1\dots y_n.xQ_1\dots Q_n$ *be an* FHP *and* $x:\rho_1\to\dots\to\rho_h\to\chi\vdash x:\sigma$ *and*

$$x:\sigma, y_1:\theta_1,\dots,y_k:\theta_k \vdash \lambda y_{k+1}\dots y_n.xQ_1\dots Q_n:\tau,$$

where χ *is a basic union in normal form and* $\uparrow(\chi)=0$. *Then* $FV(Q_1\dots Q_{min(h,n)}) = \{y_1,\dots,y_{min(h,n)}\}$.

The cases $n=0$ or $h\ge n$ are trivial. Otherwise the proof is by induction on the derivation of

$$x:\sigma,\ y_1:\theta_1,\dots,y_k:\theta_k \vdash \lambda y_{k+1}\dots y_n.xQ_1\dots Q_n:\tau.$$

If the last applied rule is $(\wedge I)$, $\wedge E)$, $(\vee I)$ or $(\rightarrow I)$ the proof follows by induction.

Let the last applied rule be $(\rightarrow E)$:

$$(\rightarrow E)\quad \frac{\Gamma \vdash xQ_1 \ldots Q_{n-1} : \vartheta \rightarrow \tau \quad y_{\pi(n)} : \theta_{\pi(n)} \vdash Q_n : \vartheta}{x : \sigma,\ y_1 : \theta_1, \ldots, y_n : \theta_n \vdash xQ_1 \ldots Q_n : \tau}$$

where $\Gamma = x : \sigma,\ y_1 : \theta_1, \ldots, y_{\pi(n-1)} : \theta_{\pi(n-1)}, y_{\pi(n+1)} : \theta_{\pi(n+1)}, \ldots, y_n : \theta_n$. If $h < n-1$ the proof follows by induction. The other case, $h = n-1$, is impossible. Since $x : \rho_1 \rightarrow \ldots \rightarrow \rho_h \rightarrow \chi \vdash x : \sigma$, rule (L) and Lemma 2.3(2) imply $z : \chi \vdash z : \vartheta \rightarrow \tau$. But this contradicts Lemma A.1(2), by the hypothesis that χ is a union type in normal form with $\uparrow(\chi) = 0$, so χ is a union of types pairwise different.

If the last applied rule is $(\vee E)$ there are different cases according to the subjects of the premises. If the subject of the third premise is the whole term:

$$\frac{t : \zeta_1 \wedge \varsigma \vdash t : \tau \qquad t : \zeta_2 \wedge \varsigma \vdash t : \tau \qquad x : \sigma,\ y_1 : \theta_1, \ldots, y_k : \theta_k \vdash \lambda y_{k+1} \ldots y_n . xQ_1 \ldots Q_n : (\zeta_1 \vee \zeta_2) \wedge \varsigma}{x : \sigma,\ y_1 : \theta_1, \ldots, y_k : \theta_k \vdash \lambda y_{k+1} \ldots y_n . xQ_1 \ldots Q_n : \tau}$$

the proof follows immediately by induction on the third premise.

Let the subject of the third premise be $xQ_1 \ldots Q_j$, for some $j \leq k$:

$$\frac{\begin{array}{c}\Gamma_1, t : \zeta_1 \wedge \varsigma \vdash \lambda y_{k+1} \ldots y_n . tQ_{j+1} \ldots Q_n : \tau \qquad \Gamma_1, t : \zeta_2 \wedge \varsigma \vdash \lambda y_{k+1} \ldots y_n . tQ_{j+1} \ldots Q_n : \tau \\ \Gamma_2, x : \sigma \vdash xQ_1 \ldots Q_j : (\zeta_1 \vee \zeta_2) \wedge \varsigma\end{array}}{x : \sigma,\ y_1 : \theta_1, \ldots, y_k : \theta_k \vdash \lambda y_{k+1} \ldots y_n . xQ_1 \ldots Q_n : \tau}$$

where $\Gamma_1 = \{y_{\pi(j+1)} : \theta_{\pi(j+1)}, \ldots, y_{\pi(k)} : \theta_{\pi(k)}\}$ and $\Gamma_2 = \{y_{\pi(1)} : \theta_{\pi(1)}, \ldots, y_{\pi(j)} : \theta_{\pi(j)}\}$.
There are two cases:

— If $k \leq h$, then also $j \leq h$. The application of rule (L) and Lemma 2.3(2) to the third premise and to $x : \rho_1 \rightarrow \ldots \rightarrow \rho_h \rightarrow \chi \vdash x : \sigma$ implies $t : \rho_{j+1} \rightarrow \ldots \rightarrow \rho_h \rightarrow \chi \vdash t : (\zeta_1 \vee \zeta_2) \wedge \varsigma$. Lemma A.1(1) gives $t : \rho_{j+1} \rightarrow \ldots \rightarrow \rho_h \rightarrow \chi \vdash t : \zeta_i \wedge \varsigma$ for $i = 1$ or $i = 2$. The induction on the first or second premise gives $FV(Q_{j+1} \ldots Q_h) = \{y_{j+1}, \ldots, y_h\}$. Now note that $FV(Q_1 \ldots Q_j) \subseteq \{y_1, \ldots, y_k\}$ and so, since each Q_i has a different head variable, $FV(Q_1 \ldots Q_j) = \{y_1, \ldots, y_j\}$, which concludes the proof of this case.

— If $k > h$, then either $j < h$ or $j \geq h$. If $j < h$, then $t : \rho_{j+1} \rightarrow \ldots \rightarrow \rho_h \rightarrow \chi \vdash t : (\zeta_1 \vee \zeta_2) \wedge \varsigma$ holds by rule (L) and Lemma 2.3(2) and the proof concludes as in the previous case.
If $j \geq h$, the result follows by applying induction to the third premise.

If the subject of the third premise is Q_j:

$$\frac{\begin{array}{c}x : \sigma, t : \zeta_1 \wedge \varsigma \vdash \lambda y_{k+1} \ldots y_n . xQ_1 \ldots Q_{j-1} tQ_{j+1} \ldots Q_n : \tau \\ x : \sigma, t : \zeta_2 \wedge \varsigma \vdash \lambda y_{k+1} \ldots y_n . xQ_1 \ldots Q_{j-1} tQ_{j+1} \ldots Q_n : \tau \\ y_{\pi(j)} : \theta_{\pi(j)} \vdash Q_j : (\zeta_1 \vee \zeta_2) \wedge \varsigma\end{array}}{x : \sigma,\ y_1 : \theta_1, \ldots, y_k : \theta_k \vdash \lambda y_{k+1} \ldots y_n . xQ_1 \ldots Q_n : \tau}$$

Since $\lambda x y_1 \ldots y_{\pi(j-1)} t\, y_{\pi(j+1)} \ldots y_n . xQ_1 \ldots Q_{j-1} tQ_{j+1} \ldots Q_n$ is an FHP, induction applies to the first premise.

The proof for the case in which the subject of the third premise is the head variable of Q_j is analogous.